

# **NJE Bridge: Configuration and Planning Reference Version 2 Release 1.0**

Sine Nomine Associates  
43596 Blacksmith Square  
Ashburn, VA 20147



---

# Contents

<b>About This Document</b> .....	1
About the NJE Bridge .....	1

---

## **Planning Your NJE Bridge Deployment** .....

<b>Before You Start</b> .....	4
What Is NJE? .....	4
Units of Work .....	4
Nodes .....	4
Users .....	6
NJE Addressing .....	7
Links/Lines .....	7
NJE Routing .....	7
General Routing .....	7
JES2 and NJE Bridge Dynamic Routing .....	8
Default Routes .....	8
What Information Do You Need to Plan Your Network? .....	8
Creating a Naming Convention .....	9
<b>Planning Your Network Topology</b> .....	10
Star Topology .....	10
Hierarchical Tree Topology .....	10
Hybrid Topology .....	11
Network Routing Tables .....	12
Printers and NJE .....	14
Default Behavior When Not Otherwise Specified .....	14
When No Node Name is Specified .....	14
When No Userid is Specified .....	14
When No Route is Specified .....	15
Special Userids and Values .....	15

---

## **Configuration Reference** .....

<b>Reading Syntax Diagrams</b> .....	18
Syntax Items .....	18
Symbols .....	18
Variables .....	18
Repetition .....	19
Required Choices .....	19
Optional Choices .....	19
Defaults .....	19
<b>General Configuration Statements (nje.cf)</b> .....	21
<blank line> (End Statement Block) .....	22
ADMGROUP (Set Unix Group Name for Administrative NJE Users) .....	23
ALIAS (Set Alternative NJE Node Names for This Node) .....	25
CERTDIR (Set Path for SSL-related Information) .....	26
CERTFILE (Set File Name of Certificate ID For This Node) .....	27
DEFAULT-ROUTE (Set Default NJE Route if No Specific Route Supplied) .....	28

DEFFORM (Set Default Form Code) . . . . .	30
EBCDICTBL (Set Path for EBCDIC to ASCII Translation Table File) . . . . .	31
FILEEXITS (Set Filename and Path for File Processing Exit Table) . . . . .	32
GROUP (Set Unix Group Name for General NJE Users . . . . .	33
IPADDRESS (Set IP Address Used in Link Signon) . . . . .	35
LICENSE (Specify PAK for This Product) . . . . .	37
LLEVEL (Set Default Log Detail Level) . . . . .	38
MSGEXITS (Set Filename and Path for Message Processing Exit Table) . . . . .	39
NAME (Set Local NJE Node Name) . . . . .	40
QUEUE (Set Queue Directory Path) . . . . .	41
ROUTE_TABLE (Set Filename and Path for NJE Routing Table - Alt Syntax) . . . . .	42
TABLE (Set Filename and Path for NJE Routing Table) . . . . .	43
USEREXITS (Set Filename and Path for File Processing Exit Table - Alt Syntax . . . . .	44
<b>Line-Specific Configuration Statements (nje.cf) . . . . .</b>	<b>45</b>
<blank line> (End Statement Block) . . . . .	46
BUFSIZE (Set Maximum Transfer Buffer Size for This Line) . . . . .	47
CERTIFICATE (Set Certificate File Base Name to Use for This Line) . . . . .	48
CONNECT (Control Connection Initiation for This Line) . . . . .	49
IPPORT (Set IP Port for NJE Traffic On Remote Host for This Line) . . . . .	50
LINE (Define A Connection to Another NJE Host) . . . . .	51
MAX-STREAMS (Set Maximum Number of Parallel Transfer Streams for This Line) . . . . .	52
NATIN (Specify "Inside" Address Value for NAT Firewall Support) . . . . .	53
NATOUT (Specify "Outside" Address Value for NAT Firewall Support) . . . . .	54
ORDER (Set Queuing Mechanism and Transmission Sequencing for This Line) . . . . .	55
PATHMGR (Enable JES2 PATHMGR Dynamic Routing Record Support) . . . . .	56
POLLTIME (Specify Wait Time When Expecting Incoming Connection Request) . . . . .	58
PRIVATEKEY (Set Private Key File Base Name to Use for This Line) . . . . .	59
RESISTANCE (Specify Line Resistance Value for PATHMGR Dynamic Routing) . . . . .	60
RLPASS (Set Line Password Expected by Remote Line) . . . . .	62
RNPASS (Set Node Password Expected by Remote Node) . . . . .	63
TCPNAME (Set DNS Name for Remote Host for This Line) . . . . .	64
TLPASS (Set Line Password Expected From Remote Line) . . . . .	66
TNPASS (Set Node Password Expected From Remote Node) . . . . .	67
TYPE (Set Type of Connection Method for This Line) . . . . .	69
<b>SSL-Related Files . . . . .</b>	<b>71</b>
SSL Basics . . . . .	71
Where Do Certificates Come In? . . . . .	71
What is an SSL Certificate and How Does it Work? . . . . .	71
How Does the SSL Certificate Create a Secure Connection? . . . . .	73
SSL Configuration Files . . . . .	73
nodename.crt . . . . .	74
nodename.key . . . . .	74
njeCA.crt . . . . .	74
Certificate Tasks . . . . .	74
Setting Up SSL with Self-Signed Certificates . . . . .	74
Generating a Self-Signed Certificate . . . . .	75
Verifying Connection Encryption . . . . .	80
Setting Up SSL Link Encryption with External Certificates . . . . .	80

---

<b>File and Message Processing Exits . . . . .</b>	<b>81</b>
--	-----------

<b>File Processing Exit (file-exit.cf)</b> . . . . .	82
File Pattern Matching Syntax . . . . .	82
Coding File Patterns And The file-exit.cf File . . . . .	82
<b>Message Processing Exit (msg-exit.cf)</b> . . . . .	86
Command Pattern Matching Syntax . . . . .	86
Interactive Message Pattern Matching Syntax . . . . .	86
Structure Of The msg-exit.cf File . . . . .	86
General Message Processing Configuration Statements . . . . .	86
Command Processing Patterns . . . . .	87
Message Processing Patterns . . . . .	88
<b>Forms, Printing and Character Set Translation</b> . . . . .	90
Printing . . . . .	90
Defining A Printer . . . . .	90
Defining A Default Printer . . . . .	90
Forms . . . . .	91
Creating a Form File . . . . .	91
SYSTEM.STANDARD (The Default System Form) . . . . .	92
Specifying Binary File Processing . . . . .	92
Character Set Translation . . . . .	92
Coding a Custom Translation Table . . . . .	93
Specifying Your Custom Table . . . . .	93
Specifying Binary Data (No Translation) . . . . .	93
<hr/>	
<b>Administrative Commands and Tasks</b> . . . . .	95
<b>Administrative Commands</b> . . . . .	96
rprint (Submit Print Jobs to Local Printing System) . . . . .	97
ucp (User Control Panel) . . . . .	99
<b>UCP Subcommands</b> . . . . .	100
EXIT (Terminate Interactive UCP Session) . . . . .	101
HELP (Display UCP Help File) . . . . .	102
LOGLEVEL (Change Logging Detail Level) . . . . .	103
QUIT (Terminate Interactive UCP Session) . . . . .	105
RESCAN EXITS (Rescan File and Message Exit Files) . . . . .	106
RESCAN QUEUE (Rescan All File Queues and Reschedule Files) . . . . .	107
RESCAN ROUTE (Reload NJE Routing Database) . . . . .	108
ROUTE (Modify NJE Routing Database) . . . . .	109
SHOW LINE (Display Defined Lines and Line Status) . . . . .	111
SHOW QUEUE (Show Status of File Transmit/Receive Processing) . . . . .	114
SHUT (Terminate All Connections Gracefully and Halt NJE Processing) . . . . .	117
START LINE (Start Inactive Line) . . . . .	118
STOP LINE (Gracefully Stop Active Line) . . . . .	120
<b>Administrative Tasks</b> . . . . .	122
Start A Line/Link . . . . .	122
Stop A Line/Link . . . . .	122
Stop A Line/Link Immediately . . . . .	122
Display File Queues . . . . .	122

---

<b>If You Have Trouble</b> .....	123
<b>If You Have Trouble</b> .....	124
Gathering Information About The Problem .....	124
Logging and Log Files .....	124
Setting An Elevated Debugging Level .....	124
What to Capture .....	124
Turning Debugging Off .....	124
Reporting A Problem .....	124
Helpful Hints While Working on the Problem .....	124

---

<b>Appendices</b> .....	125
<b>Appendix A. Other Helpful References</b> .....	126
NJE Protocol References .....	126
Connecting to IBM Implementations .....	126
RSCS Connections .....	126
VSE/POWER Connections .....	126
z/OS and JES2 Connections .....	126
z/OS and JES3 Connections .....	127
iSeries Connections .....	127
<b>Appendix B. Your Comments are Welcome!</b> .....	128

---

# Figures

1. Sample NJE Network . . . . .	6
2. Variant Addressing Forms Used with NJE on JES2 and JES3 . . . . .	8
3. Example of NJE Network in Star Topology . . . . .	11
4. Example of NJE Network in a Hierarchical Tree Topology . . . . .	12
5. Example of NJE Network in Hybrid Topology . . . . .	13
6. Sample Network Using Default Routes . . . . .	29
7. Intermediate Certificate Chain . . . . .	72
8. Sample nje.cf Configuration for IPv4 SSL link to CTS6NJE . . . . .	75
9. Patch to the openssl Configuration File . . . . .	77

# **Tables**

1.	Symbols in Syntax Diagrams . . . . .	18
2.	Key SSL Statements in nje.cf . . . . .	75
3.	General Configuration Statements in file-exit.cf . . . . .	83
4.	Columns and Pattern Entries in file-exit.cf . . . . .	84
5.	General Message Processing Configuraiton in msg-exit.cf . . . . .	87
6.	Command Message Processing Configuration in msg-exit.cf . . . . .	87
7.	Interactive Message Processing Configuration in msg-exit.cf . . . . .	88
8.	Logging Detail Levels . . . . .	103
9.	SHOW LINE Parameters . . . . .	111
10.	Line Status Values . . . . .	112
11.	SHOW QUEUE Response . . . . .	114
12.	File Dispatch Code Values . . . . .	115
13.	File Processing States . . . . .	115



---

## About This Document

User command references and usage notes are located in the End User Command Guide.

This document describes the complete reference for configuration statements and routing definition for Sine Nomine's full NJE Bridge product. The document describes the general and line-specific configuration statements, network planning guides and exits for file and message processing.

---

## About the NJE Bridge

The NJE/IP Bridge is a product designed to enable TCPNJE connections between traditional IBM NJE hosts and open/discrete systems. It supports communication between any supported platforms (broadly, almost any system with a POSIX.1 interface) and RSCS, JES2, JES3, VSE/POWER, or other NJE/IP Bridge systems.

The NJE/IP Bridge is designed to allow file transfer and remote job submission to traditional mainframe hosts from open systems platforms, as well as to enable submission to open systems platforms from mainframe hosts. It is particularly useful for enabling printing from mainframe hosts to printers in an open systems environment and unattended file transfers between any combination of NJE-connected systems. The NJE/IP Bridge provides a mechanism for batch job control, and a general mechanism for messaging between users/services/virtual machines on either end of the link, which makes it well-suited for the design of a distributed processing system.

The NJE/IP Bridge provides:

- The ability to define TCP (version 4 and version 6) links point-to-point connection with or without SSL protection
- A store-and-forward file transfer service
- Interactive messaging
- The ability to take arbitrary action via user-defined exits on receipt of a message or file; this can include mapping of "pseudo-users" to particular services.
- If the system is equipped with NQS, the ability to accept remotely-submitted jobs, execute them, and return output to arbitrary NJE destinations.

A product activation key (PAK) is required for the product to function. Please contact Sine Nomine Associates before installation to obtain a PAK for your system.



---

# Planning Your NJE Bridge Deployment

---

# Before You Start

---

## What Is NJE?

Network Job Entry (NJE) is a set of protocols and software implementations defined by IBM that provide the ability to transfer files and interactive messages between hosts implementing NJE applications. NJE implementations can run on all of IBM's major operating systems, and implementations of varying degrees of completeness exist for non-IBM systems, such as Sine Nomine's NJE Bridge (which provides a full implementation). NJE allows the user to perform unattended file transfers to remote systems where files and messages can be delivered directly to users, or be processed by a batch job monitor if the system supports one, or processed by an application that responds to incoming files and requests with data requested by a user. Finally, NJE provides the ability to send commands related to the operation of the network or system to be executed at a remote node, returning the results of those commands to another node in the NJE network.

NJE relies on some basic concepts and ideas to operate. The next few sections discuss these basic concepts.

## Units of Work

An NJE unit of work (sometimes referred to in the IBM documentation as a "transfer unit") is a discrete unit of information that is transmitted across the network. An NJE transfer unit can be either an NJE job or a nodal message record (NMR).

An NJE job is a transfer unit that contains data to be processed at another node in the NJE network. It begins with a job header, is followed by data, and ends with a job trailer. The type of data contained in the NJE job further defines the type of NJE job. The data between the job header and job trailer can be either SYSIN or SYSOUT data. An NJE SYSIN job is an NJE job that contains JCL for a job and may have one or more SYSIN data sets. An NJE SYSOUT job is an NJE job that contains one or more SYSOUT data sets. Each SYSOUT data set is preceded by a data set header.

A nodal message record (NMR) is a unit of work that begins with an NMR header and is followed by message text. The message text can be either a message or a system command.

## Nodes

A node is a system or complex of systems that supports NJE connectivity. A node in a NJE network can be another complex or system within a single location or it can be a complex that resides in a remote location. Each node that a complex can access must be identified to other complexes by a unique NJE node name.

The NJE node name appears in job headers, data set headers, and NMRs. Do not confuse link or line names with the node name, they are two separate entities.

Each node in the network can do the following with an NJE transfer unit:

- Transmit

The node packages the NJE transfer unit and transmits it to another node.

- Receive

The node recognizes the NJE transfer unit, receives, and stores it.

- Store-and-forward

The node accepts the NJE transfer unit, stores it, and schedules it to be forwarded to another node.

The IBM documentation for NJE uses the following terminology for the nodes that comprise an NJE network.

- Originating Node

An originating node is the node where the user submitted the request to transmit the data to another complex.

- Intermediate Node

An intermediate node is a node that lies in the path of either the:

- Originating node and execution node
- Execution node and the destination node

An intermediate node both receives and transmits the NJE transfer unit to the next node in the path of the target node.

- Target Node A target node is the node where an NJE job or NMR is received and will either be executed or be processed. The target node can be either a:

- Destination Node

A destination node is a node that receives and processes:

- An NJE SYSOUT job. A node processes an NJE SYSOUT job by printing or punching the SYSOUT data set.
- A message contained in an NMR.

When an NJE transfer unit reaches its destination, it may or may not be processed as the user intended, depending on the facilities available at that node. NJE protocols allow the destination node to reject files that it cannot process or perform other system-dependent actions.

- Execution Node

An execution node is the node where:

- JCL contained in an NJE SYSIN job executes. The node packages the SYSOUT data sets created by the SYSIN in an NJE SYSOUT job and sends the NJE job to the destination node.
- A command contained in an NMR is processed. The node packages the messages that are a result of the command in an NMR and sends the NMR to the destination node.

The execution node may not necessarily be the destination node. If, for example, a user submits a job specifying that the job execute at one complex and job's output be printed at a different complex then the complex where the job runs is the execution node and the complex where the SYSOUT prints is the destination node. If no SYSOUT destination is specified, then the origin node and destination node are the same by default.

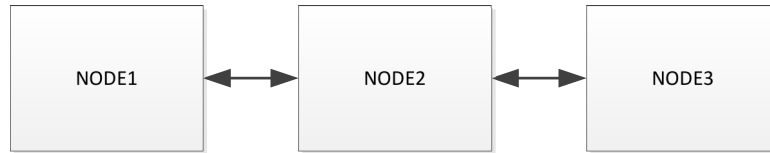


Figure 1. Sample NJE Network

Figure 1 on page 6 illustrates the different types of nodes in an NJE network. The network is composed of 3 nodes. If a user submits a job at node 1 to be executed at node 3:

- Node 1 is the originating node because that is where the user submitted the request.
- Node 2 is the intermediate node, because it is in the path of the destination node. Node 2 receives the data, stores it, then forwards it to the next node in the path of the target node.
- Node 3 is both the execution and destination node because it is the node that the user specified as the target.

To transmit an NJE transfer unit to a complex other than the user's installation (a remote node), the user issues a command or submits a job specifying a destination node name. The destination node can be either directly- or indirectly-connected to the originating node. In the network depicted in Figure 1, if NODE1 is the originating node, NODE2 is a directly-connected node to NODE1, and NODE3 is an indirectly-connected node to NODE1.

## Users

The IBM documentation refers to several types of user.

- Originating User

An originating user is the user that submits an NJE transfer unit at the originating node. The originating user submits the NJE transfer unit at an operator console, terminal, or an RJE workstation. An NJE transfer unit may originate from another NJE transfer unit.

- Destination User

A destination user is a user or device (printer or punch) that is the target of an NJE SYSOUT job. Printers at a node are destination users.

- Notification User

A notification user is the user who receives messages that notify the user of the status of the NJE transfer unit.

- Accounting User

An accounting user is the user that receives the notification of the amount or cost of system resources used in processing an NJE transfer unit.

## NJE Addressing

NJE protocols route an NJE job to the specified destination based on addressing information supplied by the originating system. The format of specifying that information depends on the NJE implementation in use; some systems use the common user@node convention present in modern electronic mail addresses, but some implementations use variant forms that integrate with that node's JCL. Figure 2 on page 8 shows some alternative variations implemented by z/OS and JES2 or JES3. The form using only the nodename is not recommended for new installations; if the destination is a NJE Bridge node, the destination is assumed to be Node.SYSTEM and is processed according to the file processing exit entry in file-exit.cf. The destination node routes the job to the remote or user at the destination node.

The JES2 and JES3 NJE implementations support defining an abstract identifier (referred to as a DESTID) that indicates a node/userid combination. When possible, we recommend use of DESTIDs to indicate specific destinations, particularly when used to indicate printers or applications defined using the file exits. Use of DESTIDs allows the actual node and userid combination representing a function to change in the future without updating JCL or processes.

## Links/Lines

Links and/or lines (the terms are used interchangeably) connect nodes together. An NJE network is a fully-specified tree. Each node must know the next link to use to reach each and every other node; unlike TCP/IP networking, an unknown destination address is an error, rather than a problem for the next node in the chain. In contrast to Ethernet, which can be thought of as a bus topology, NJE is a connected set of point-to-point links. The definition of these links and their logical topology provides the heart of NJE routing and message transmission.

Each link provides a point-to-point connection between two nodes. Each node name is 1-8 characters, and each node name must be unique. This does restrict the maximum number of nodes compared to an arbitrarily-nested scheme such as the DNS. In practice, each site will configure the NJE IP Bridge for its own internal use, and use other TCP/IP protocols to talk to other sites, so this does not present a significant limitation.

## NJE Routing

For every node known to NJE (which is every node in the entire network) NJE must know which link traffic for that node should flow through. Although NJE is a fully specified tree, it is not necessary for each node to know the complete routing path for every other node; instead, only the next hop for traffic for a given node must be specified.

## General Routing

Each node must have a route for every other NJE node in the network specified in the NJE routing table on that node. Routes may be directly-connected or indirectly-connected.

Each directly-connected node has an implicit route: the link that connects the two nodes. All other valid nodes need route statements instructing NJE how to route traffic to those nodes. NJE has the concept of a default route, down which all traffic not explicitly specified for a particular link will flow. It is presumed that the recipient node will have additional routing information, should it not be the ultimate destination, so that it can choose the appropriate link for the messages it relays for further transmission. It is advisable, if at all possible, to make the nodename (or an alias for the real nodename) either the DNS hostname, or a truncated version thereof.

---

Destination = Node

Destination = (Node.Remote-id)

Destination = (Node.Userid)

Destination = (DESTID)

---

Figure 2. Variant Addressing Forms Used with NJE on JES2 and JES3

## JES2 and NJE Bridge Dynamic Routing

For JES2 (arguably the most advanced NJE implementation), additional NMR records provide the ability for connections and routing information to be created dynamically based on connection weighting information and additional records generated by the JES2 Network Path Manager (NPM) application when nodes connect and disconnect from the NJE network. Until the release of the NJE Bridge, only JES2 could take advantage of these records - the NJE Bridge included code to extend this function to non-JES2 nodes.

A JES2 node uses the NPM to establish and manage a connection. NPM notifies other nodes about connections and disconnections to the network and maintains line status information. In controlling network traffic, NPM relies upon a (programmer-specified) line resistance value and the use of connection event sequence (CES) values taken from the system TOD clock. Connection status information is traded between Network Path Managers (NPMs) through Add Connection and Subtract Connection records, dynamically building topology and routing information databases.

## Default Routes

Some NJE implementations implement the concept of a "default" route as a catch-all if no more specific route is present. The default route is used if there is no entry in the routing database for a destination node name. For nodes that have few (or one) links, this allows all traffic to be passed to the link identified as the default route. If additional links are added, specific routes can be added to the local routing database, but all nodes not specified in the local routing database still flow through the default route.

---

## What Information Do You Need to Plan Your Network?

To plan your network you need the following pieces of information:

- A list of nodes to be included in the network.
- A connection diagram of how you wish traffic to flow. See "Planning Your Network Topology" on page 10 for a discussion of planning network topology and connections. A block-diagram sketch is frequently helpful in determining routing tables.
- A list of destination users that require special processing (eg, printers, applications that should automatically process files or messages).
- Routing table information for each node. The routing table includes the next hop for each node defined.

Your list of network nodes should include the following items for each node:

- The destination's NJE nodename.
- Its hostname in the DNS (preferred), or its IP address.



- The TCP port on which it listens.
- Expected maximum transmit buffer size value (BUFSIZE)
- Expected maximum # of active streams (MAX-STREAMS)

The BUFSIZE and MAX-STREAMS values must match in the connection definitions at each end of the links between nodes, although as long as the values match, any values can be used. If SSL-protected connections are needed, additional information on certificates and keys to be used to encrypt the session will be required. Refer to “SSL-Related Files” on page 71 for more information on SSL sessions.

---

## Creating a Naming Convention

Any string of 1 to 8 characters made up of:

- Uppercase Roman alphabetic letters from A to Z
- Numerals from 0 to 9
- Symbols from the set (-\$\_)

can be used in NJE node names, however symbols can cause problems with some implementations and should be avoided if possible. To ease remembering node names, you should develop a convention for node naming, as the 8 character nodename length can lead to difficult to remember destinations.

One popular convention is to use a geographically-based name using the two-letter ISO country code as the first two letters of the node name, followed by a 3 letter region id and a 3 character serial number, e.g. USSWA001 for a system located in the United States Southwest.

Some sites extend this convention by using link names that identify a branch of the overall NJE network that can be reached via that link; using our American Southwest example above, some international companies might have a node US that connects all US nodes. The US node may have connections to all the US regional nodes, eg USSWA (southwest), USNWA (northwest), USMWA (mid-west), etc. , and regional nodes may have direct connections to all the endpoint. This approach allows a majority of nodes to use default routes for most nodes; only the regional and national nodes need more complete routing tables.

Regardless of the convention chosen, you should be consistent in applying the convention. Consistent use of a convention will dramatically simplify network operation.

---

# Planning Your Network Topology

There are two major topology designs used in NJE networks: a star topology, and a hierarchical tree topology. Both approaches have advantages and disadvantages. The following sections describe both approaches and their advantages. A third hybrid design (a combination of the two approaches) is sometimes used in networks with diverse NJE implementations where a majority of nodes do not support the JES2 and NJE Bridge Network Path Manager application.

---

## Star Topology

A star topology configures all end systems to connect to one or two central systems directly, as shown in Figure 3 on page 11. Links are defined from the central systems to each end node directly. End systems send all NJE work units directly to the central node which routes the work unit to the correct destination node.

The major advantages of a star topology is simplicity in routing and connectivity management. Since a link definition from the central node to a leaf node generates a implied route in the NJE routing database, all leaf nodes can be configured with a default route to the central node and no routing databases need be maintained on leaf nodes (assuming the leaf node supports default routing). Transporting NJE traffic via TCP/IP removes most of the requirements for routing traffic through intermediate nodes in a modern deployment (all nodes can directly reach the central node without intervening hops).

The major disadvantage of a star topology is that some operations (such as defining new nodes) require restarting the NJE implementation at the central sites, disrupting file and message flow to all nodes. If the central nodes are not JES2 or NJE Bridge nodes, traffic must be rerouted manually if a central node fails (the use of Network Path Manager-capable nodes as central nodes is highly recommended, in that those nodes will adjust NJE routing automatically as required).

A second disadvantage of the star topology is that all files and traffic must flow to the central node and back, even if the destination node is at the same location as the sending node. If the central node is in the same location, this may not significantly impact your network performance, however if WAN links are involved, the amount of bandwidth used in repeatedly transmitting data to and from the central nodes may introduce significant processing delays for files. (This scenario is often the most compelling argument for a hybrid or hierarchical topology).

If a majority of the nodes in your NJE network will not be implemented on JES2 or NJE Bridge systems, the star topology is the recommended topology for new deployments.

---

## Hierarchical Tree Topology

The second common topology (and the one used in many classical NJE deployments where traffic was carried over actual physical telecommunication lines from point to point) is a hierarchical tree, where the network is structured as multiple levels of a tree with leaf systems attached to regional nodes, which attach to larger regional systems. Figure 4 on page 12 shows a small example of a tree structure in a NJE network.

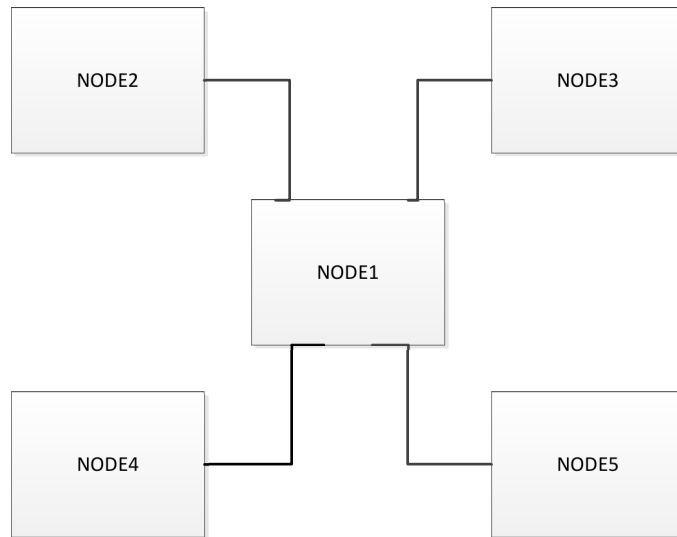


Figure 3. Example of NJE Network in Star Topology

The primary advantage of the hierarchical tree topology is when groups of NJE nodes primarily communicate with each other, but occasionally need to communicate outside their local group, possibly via an expensive or resource-constrained connection. NJE traffic between group members flows from leaf nodes to the designated "concentrator" node for the group, and need no specialized routing for traffic between nodes in the group. Files and messages intended for nodes in other groups are routed and queued as required. Interruptions in service in other areas of the network do not affect traffic between nodes within a group. The network can use this structure at an arbitrary number of levels, as needed, and interconnections between different levels of the tree are possible (although complex).

The disadvantage of the hierarchical tree topology is the need to maintain routing tables at each level of the tree. Only nodes within a level in the tree can take advantage of the implied routes created by direct links; all other nodes present in the network must have explicit routes defined on most of the hosts. For networks with more than 50-60 nodes this can be a substantial task.

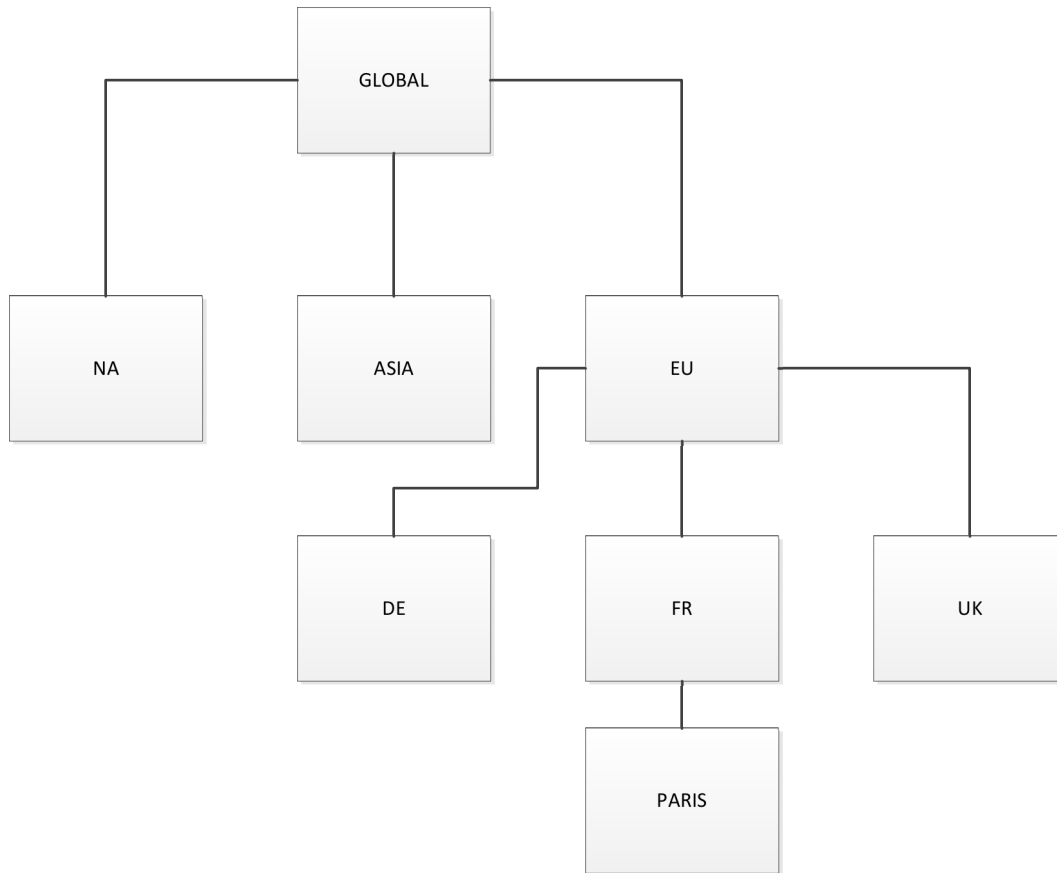
The hierarchical tree topology is most frequently used where a TCP implementation is replacing an existing NJE implementation and infrastructure is already in place to create and distribute routing tables, or where there are several logical divisions of the network connected by bandwidth-limited connections (the most common case is a multinational company with segments of the network on multiple continents separated by restricted-capacity or high-latency WAN links).

---

## Hybrid Topology

Some organizations implement a composite of the two major topology types: a core mesh of JES2 or NJE Bridge nodes interconnecting star networks for geographic regions. Figure 5 on page 13 is an example of a hybrid configuration.

The advantage of the hybrid topology gains the benefit of the JES2/NJE Bridge NPM dynamic routing capability for the core mesh nodes with the locality of a shallow hierarchical tree topology. The core JES2/NJE Bridge nodes can be interrupted or partitioned



---

Figure 4. Example of NJE Network in a Hierarchical Tree Topology

by connectivity failures, but the remainder of the network will dynamically reconfigure to work around a failed node or connection with minimum human intervention. End nodes can use default routes (if supported), and if configured, the core can act as a fully meshed network with multiple redundancies.

Configuring a hybrid topology can be quite complex; if you wish to implement this topology, please contact SNA support for assistance in planning the deployment.

---

## Network Routing Tables

To some extent, NJE routing is similar to IP routing in that each node does not need to know the entire path from originating node to destination node, but only the link needed to reach the next node in the chain. NJE is different from IP routing in that all possible destination nodes must be known at each node along the way or the NJE work unit will be treated as an unknown destination at the node that does not have a route for the destination node (and handled according to the defined procedure at that node).

Routes are classified as two types:

- Directly connected systems

Each link between two NJE nodes automatically defines a route at the two endpoint of the link between the nodes, e.g., if a link connects NODE1 with NODE2, NODE1

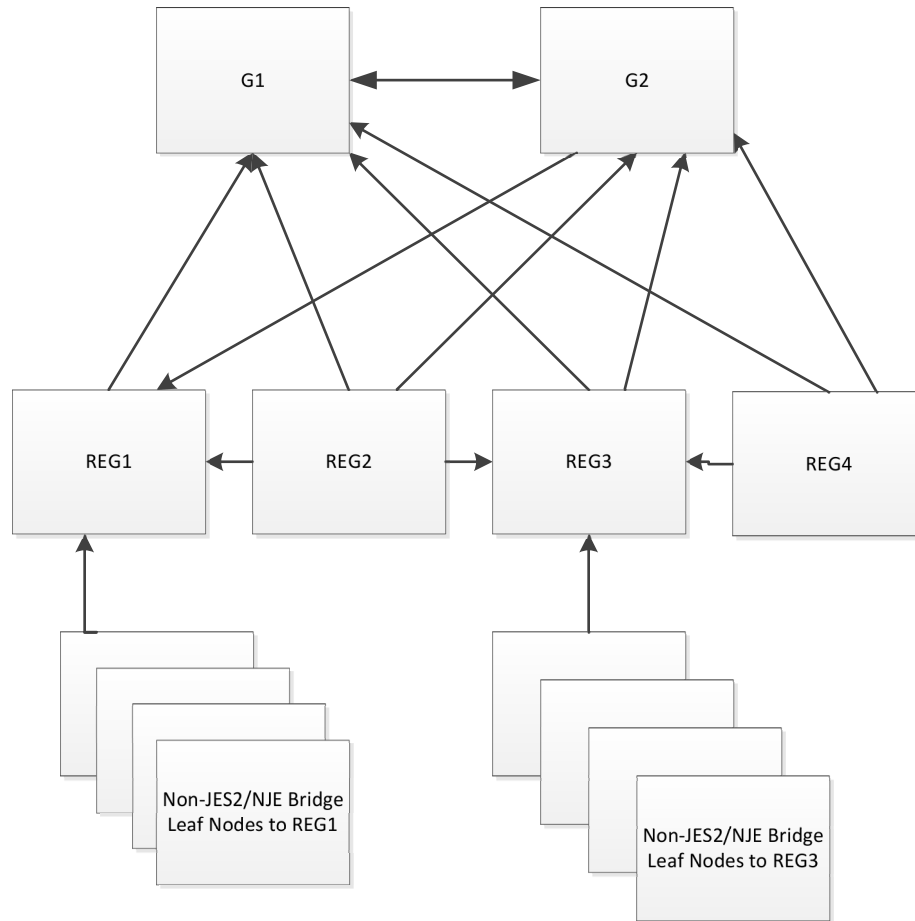


Figure 5. Example of NJE Network in Hybrid Topology

and NODE2 automatically have a route to reach each other when the link between the two sites connects.

- Indirectly connected systems

Indirectly connected systems are reachable via a connection to an intermediate node (refer to Figure 1 on page 6 for a diagram showing directly connected and indirectly connected systems).

A ROUTE statement (or equivalent) for each possible destination node must be inserted on the originating node indicating which link is to be used to send traffic to the destination node. Traffic is passed from one node to the next until the destination node is reached. To illustrate, if we use the network shown in Figure 1 on page 6, NODE1 is directly connected to NODE2 and needs no route to communicate with NODE2. NODE3 is indirectly connected to NODE1 via NODE2, so a ROUTE statement on NODE1 is needed to indicate that NODE3 can be reached via NODE2. The full routing table needed would appear similar to (assuming links are named according to the name of the NJE host on the remote end of the link):

- NODE1
  - ROUTE NODE3 TO NODE2
- NODE2

No routes are needed, as all hosts are directly connected to NODE2

- NODE3

ROUTE NODE1 TO NODE2

Note that there must be routes covering the path both from the originating node to the destination node, AND from the destination node back to the originating node. The paths used do not have to be the same in both directions, but both nodes and the intervening nodes must have some entry for the originating and destination nodes.

It is generally wise to draw a diagram of the network and work out the routing for each node while planning the network. Each NJE implementation uses different methods and locations for specifying routing information. Refer to the descriptions of the ROUTE command (“ROUTE (Modify NJE Routing Database)” on page 109) for more information on defining routing tables with the NJE Bridge.

---

## Printers and NJE

In general, printers appear as destination userids at the destination node, e.g. a printer link historically appeared as an NJE destination of nodename.linkname for print and punch SYSOUT files. The NJE Bridge uses the file processing exit to pass files to the local printing system for processing (on the Unix implementation, the preferred print system is the Common Unix Printing System (CUPS) although the Berkeley and System V printing systems can be used with reduced functionality). The print form specified in the NJE job header determines how the print file is processed. Refer to “Printing” on page 90 for more information on printer setup in the NJE Bridge.

---

## Default Behavior When Not Otherwise Specified

While IBM does not codify the behavior of specific conditions in the NJE specification, some conventions have developed for use of certain userid values and actions. The following sections describe what the NJE Bridge implementation does when the named condition occurs; consult the documentation for your NJE implementation to determine what will happen in that implementation.

### When No Node Name is Specified

If no node name is supplied in the NJE header or by the NJE utilities on an originating system, any userid value specified is assumed to be on the current system.

The NJE Bridge implementation does not permit this condition to be created (all utilities and daemons explicitly add the local NJE node name when a transmission is initiated).

### When No Userid is Specified

If no userid is specified in the NJE header or by the NJE utilities on an originating system, the processing is implementation-dependent, and may be different depending on whether the file is incoming or outgoing, or the message is incoming or outgoing.

For files, if the file is outgoing and the file has not reached its final destination, the file is passed on unmodified (files in transit to other nodes are not modified while still in transit). If the local system is the final destination node, and the userid field in the NJE file header is blank, the file is spooled to userid SYSTEM. An entry in file-exit.cf can specify additional processing if needed.

For messages, if no userid is specified, the message is dropped.

## When No Route is Specified

If no default route is specified, or no route to the next node is available for the destination node, files destined for unknown nodes are transferred to the node administrator and a message is returned to the originating NJE address indicating the file was not delivered.

In the NJE Bridge implementation, unroutable files are delivered to the POSTMAST id identified in the `nje.cf` file.

## Special Userids and Values

In the IBM implementations, the userid `SYSTEM` is normally given special treatment. It typically is interpreted as "system default", and (depending on the type of file) is processed in particular ways, to wit:

- If the file is a print or punch file, the file is sent to the default system printer tagged as class A, form STANDARD. For the NJE Bridge, this can be modified by the `DEFFORM` statement (see “`DEFFORM (Set Default Form Code)`” on page 30 for details).
- If the file is a card deck/job, the file is sent to the input card reader of the default batch system. In the NJE Bridge, this processing is controlled by the default entry at the bottom of `file-exit.cf`

**SYSIN:** The `SYSIN` userid is treated similarly to the card deck processing for userid `SYSTEM`. It is normally used for JES2 and JES3 systems, as JES2/JES3 use this term as the default job input device. If NQS is installed, the NJE Bridge implementation uses `SYSIN` as the default input to the NQS queueing system.





---

# Configuration Reference

---

# Reading Syntax Diagrams

To read a syntax diagram for entering a command, follow the path of the line. Read from left to right and from top to bottom.

---

## Syntax Items

Syntax items, such as a keyword or a variable, can be:

- On the line (required element)
- Above the line (default element)
- Below the line (optional element)

---

## Symbols

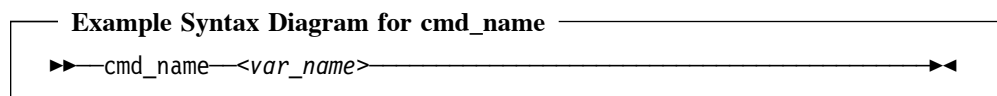
Enter these symbols exactly as they appear in the syntax diagram.

Table 1. Symbols in Syntax Diagrams	
Symbol(s)	Description
*	Asterisk
{ }	Braces
:	Colon
,	Comma
=	Equal Sign
-	Hyphen
( )	Parentheses
.	Period
	Space
"	Quotation mark
'	Single quotation mark

---

## Variables

Italicized lowercase items such as *<var\_name>* indicate variables. In this example, you can specify a *<var\_name>* when you enter the *cmd\_name* command.



---

## Repetition

An arrow returning to the left means that the item can be repeated. A character within the arrow means that you must separate repeated items with that character.

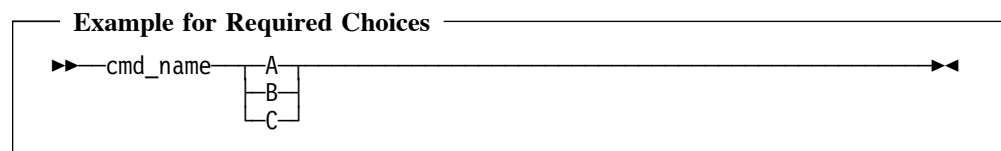
A footnote (1) by the arrow refers to a limit that tells how many times the item can be repeated.

---

## Required Choices

When two or more items are in a stack and one of them is on the line, you must specify one item.

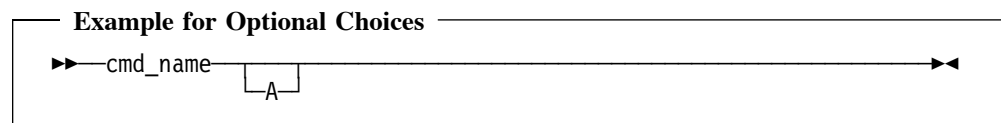
In this example, you must choose A, B, or C.



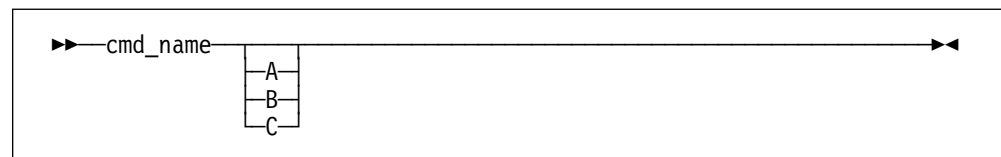
---

## Optional Choices

When an item is below the line, that item is optional. In the first example, you can select A or nothing at all.



When two or more items are in a stack below the line, all of them are optional. In the second example, you can choose A, B, C, or nothing at all.



A stack of items followed by an arrow returning to the left indicates that you can select more than one item, or in some cases, repeat a single item.

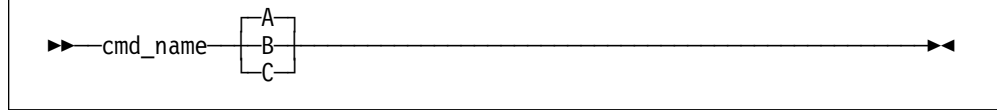
---

## Defaults

Defaults are above the line. The default is selected unless you override it, or you can select the default explicitly. To override the default, include an option from the stack below the line.

In this example, A is the default. Select either B or C to override A.

**Example for Default Values**



If you have trouble interpreting the syntax diagrams, please contact SNA support or submit a reader's comment form.

---

## General Configuration Statements (nje.cf)

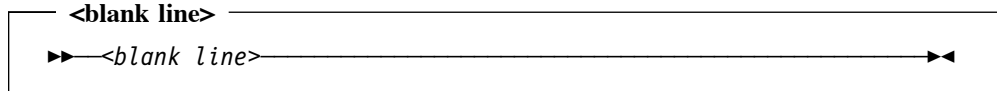
This section of the document contains the general configuration statements in the `nje.cf` file that affect the behavior of the entire product. Changes in these statements should be carefully considered, as invalid values will cause the product to not function correctly.

---

## <blank line> (End Statement Block)

### Purpose

### Format



### Parameters

#### <blank line>

A blank line terminates the current statement block.

Note that the statement is NOT visible - it is a empty line.

### Usage

Insert a blank line following each group of statements. If used at the end of the general configuration statements, it terminates the general configuration section.

### Examples

An example of the use of a blank line might appear similar to:

```
<blank line>  
LINE 10 SNARF1  
.  
. (other line definition statements)  
.  
<blank line>  
LINE 11 SNAFU  
.  
. (other line definition statements)  
.  
<blank line>
```

### Comments

1. On all platforms, a blank line is constituted of no characters followed by the platform line-end sequence (LF on Unix, default on other platforms).

---

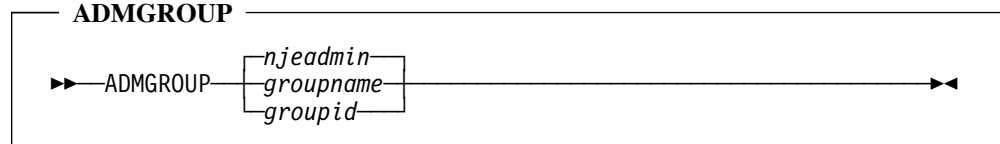
# ADMGROUP

## (Set Unix Group Name for Administrative NJE Users)

### Purpose

The ADMGROUP statement identifies a Unix groupname or id that users expected to be able to use the NJE Bridge administration tools should be a member of. Users expected to use the ucp utility should be members of this group.

### Format



### Parameters

#### njeadmin

If this statement is not specified, njeadmin is the default administrative group.

#### groupname

Unix groupname that users of the NJE Bridge administrative tools should belong to. This entry must exist in /etc/groups.

The installation process creates a group named njeadmin

#### groupnum

A numeric Unix group number that users of the NJE Bridge userpace tools should belong to.

Note that while bare numeric entries work, good system administration practice should use group names in /etc/groups to avoid future conflicts with new groups, or having to update large numbers of userid entries in /etc/groups at a later date.

### Usage

You should restrict use of the NJE administrative tools by placing them in a separate group and assigning users to that group on an individual basis.

### Examples

An example of allowing all users of the system who are members of group njeadmin to use the NJE Bridge administrative utilities might appear similar to:

```
GROUP njeadmin
```

An example of specifying the group by number might appear similar to:

```
GROUP 1101
```

### Comments

1. This statement is optional. The installation process creates a group named njeadmin as part of the install. If no ADMGROUP statement is found, the software uses the njeadmin groupname.
2. Membership in this group can modify the configuration of the NJE Bridge, and should be limited to system administrators and other highly trusted users. This entry should not reflect a widely-used group name or id.

3. Use of numeric group ids is discouraged, as it frequently causes later conflicts with unexpected results.



---

# ALIAS

## (Set Alternative NJE Node Names for This Node)

### Purpose

The ALIAS statement allows specification of additional NJE node names that are equivalent to the primary NJE node name specified on the NODE statement.

### Format

```
ALIAS  
▶ALIAS altnode◀
```

### Parameters

#### **altnode**

Additional 1-8 character NJE node name that should be considered to be equivalent to the primary node name.

### Usage

This keyword is primarily intended to be used as a transitional measure if a NJE node needs to be renamed. It assumes that userids do not change, i.e. that user@NODE and user@ALIAS are equivalent.

### Examples

An example situation where ALIAS node names might be used is when a NJE node name has been in use for a period of time, but an acquisition or merger of one company with another company requires renaming a node to match the naming conventions of the company being merged into. The node's new primary NJE node name would be listed on the NAME statement with the old name listed on the ALIAS statement. The statements in nje.cf might appear similar to:

```
NAME NEWNAME  
ALIAS OLDNAME
```

### Comments

1. This parameter is optional. If not specified, undefined node names are tested against the NJE routing tables, and if no entry is found, the file is transferred to the post-master id.
2. ALIAS node names must be valid NJE node name values. NJE node names may contain one (1) to eight (8) letters from A to Z, numbers from 0 to 9, and \$ (although use of special characters is discouraged as some other non-IBM NJE implementations do not permit special characters).
3. If your NJE node has multiple DNS entries, it is desirable to have all the possible names (truncated to valid NJE node values) represented on an ALIAS statement. This suggestion is not enforced by the NJE Bridge code, but often saves confusion.
4. If you list multiple nodes in an ALIAS statement, the NJE node names listed should also be listed as CNAME records in the DNS for this IP host. The NJE Bridge code does not enforce this suggestion, however it often saves confusion in that other nodes can use this information in place of hard-coding IP addresses in LINE statements. See "TCPNAME (Set DNS Name for Remote Host for This Line)" on page 64 for additional information on using the DNS name to refer to a NJE node name.

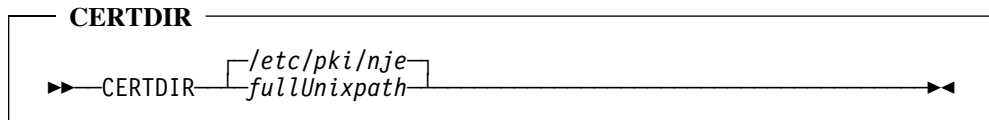
---

## CERTDIR (Set Path for SSL-related Information)

### Purpose

The CERTDIR global configuration statement indicates the full Unix path of the directory used to store SSL certificate and key information used by this node.

### Format



### Parameters

#### **/etc/pki/nje**

Default path if no CERTDIR statement is specified. This value should not be changed without discussion with SNA support.

#### **fullUnixpath**

Full Unix pathname to directory holding certificates and keys for SSL processing.

### Usage

The default directory should not be changed without careful consideration. If you need to change this value, please contact SNA support.

### Examples

An example of moving the certificate/key directory might appear similar to:

```
CERTDIR /etc/pki/nje
```

### Comments

1. See “SSL-Related Files” on page 71 for more information on SSL processing with the NJE Bridge.

---

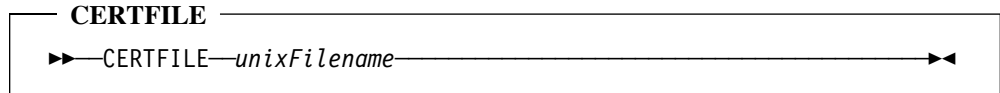
# CERTFILE

## (Set File Name of Certificate ID For This Node)

### Purpose

The CERTFILE global configuration statement indicates the file used to specify the identity of this node.

### Format



### Parameters

#### **unixFilename**

Filename of Unix file containing the identity certificate for this host.

### Usage

This file contains the certificate used to identify this host. One file should be sufficient for most implementations.

### Examples

An example of the certificate file name might look similar to:

```
CERTFILE mycert.pem
```

### Comments

See “SSL-Related Files” on page 71 for more information on SSL processing with the NJE Bridge.

---

## DEFAULT-ROUTE

### (Set Default NJE Route if No Specific Route Supplied)

#### Purpose

The DEFAULT-ROUTE statement provides an NJE route used if no more specific route is supplied either by a connected link or a routing statement created in the NJE routing database by using the ucp ROUTE command.

#### Format

```
DEFAULT-ROUTE linkname
```

#### Parameters

##### **linkname**

Any valid link name connecting this NJE node to another NJE node. Files are queued on the named link (and if active, transmission begins immediately).

#### Usage

The DEFAULT-ROUTE statement is particularly useful for leaf nodes that have only a single connection to other nodes (and thus no reason to maintain a full routing table -- all traffic is sent to the single connection if not intended for a local user). However, the DEFAULT-ROUTE statement is also useful for nodes with multiple connections in that it allows network definitions to be minimized to only connections downstream of a particular node, e.g., you need only to define routes for nodes that are beyond your local connected nodes. All other traffic can be sent to the default route link and maintaining definitions for the entire network in multiple locations can be avoided.

#### Examples

Assume that your network is similar to Figure 6 on page 29. Node F has direct connections to nodes B, C, D and E. Node A has connections to node C and E. Node Q has only one connection to node D.

Node Q can use the DEFAULT-ROUTE D statement to indicate that all traffic to other NJE nodes is via node D. Node F needs explicit ROUTE statements to reach node A and Q similar to:

```
ROUTE A TO C  
ROUTE Q TO D
```

The other nodes are directly connected to node F, and thus have implied routes for traffic to that node.

This parameter is often used in nodes intended to provide print or archival services in that those nodes are typically leaf nodes and the overhead needed to maintain full routing tables is unnecessary.

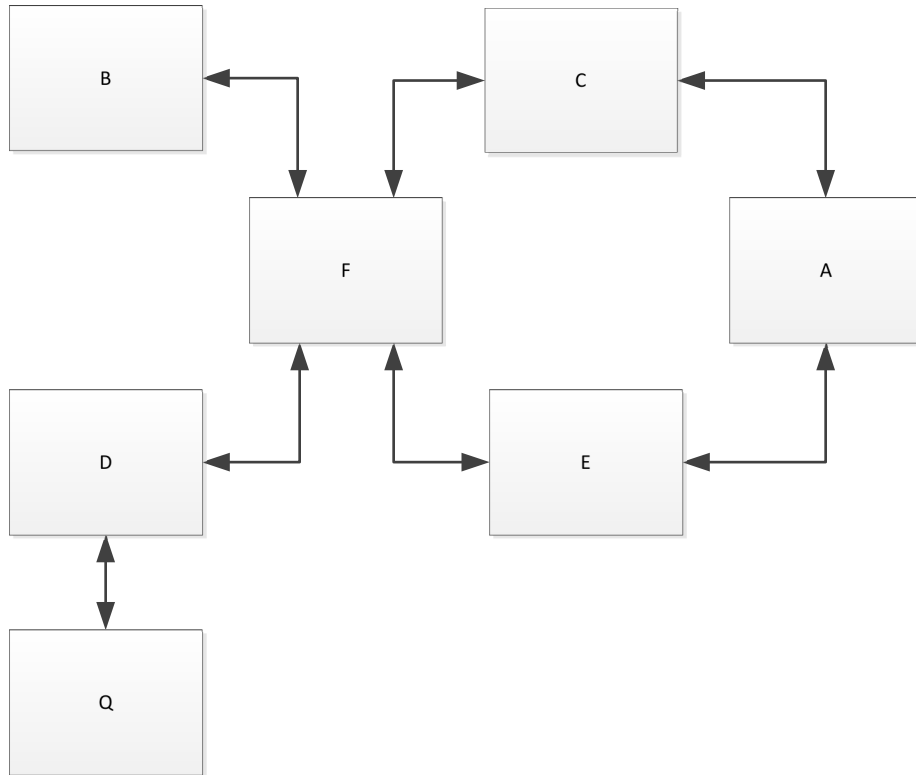


Figure 6. Sample Network Using Default Routes

### Comments

1. For a more detailed discussion of NJE routing and routing tables, see “Network Routing Tables” on page 12.
2. A LINE statement matching the value of this parameter must exist if used. All traffic that does not have a more specific route is sent to the link specified.

---

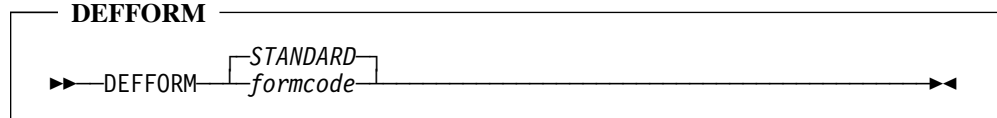
# DEFFORM

## (Set Default Form Code)

### Purpose

The DEFFORM statement specifies what form code is assigned to a print or punch file if the file does not arrive with a form code assigned. The form code must exist in `/etc/nje/forms` and can be parsed by `/usr/local/bin/rprint` to simulate sophisticated form handling. See “Forms, Printing and Character Set Translation” on page 90 for more information on defining and managing forms codes.

### Format



### Parameters

#### STANDARD

Default form code if no replacement is specified.

#### formcode

The name of a form code. The specified form must exist in `/etc/nje/forms`, be non-null and be assigned to user SYSTEM in `/etc/nje/forms`.

### Usage

This parameter is usually needed when working with NJE installations on JES2 or JES3 where the sending system has not specified a default form in the JES startup options. The default value is usually acceptable for most cases.

### Examples

Setting the default form to SYSTEM.PORT might appear similar to the following:

```
DEFFORM PORT
```

### Comments

1. Note that the syntax for form names is `user-or-link.formname`. User SYSTEM is special and is normally used to indicate system defaults. The SYSTEM.STANDARD form supplied with the software has no special processing attached to it, and the name is simply inserted into the NJE file header as the file is received.
2. This statement only modifies files which are to be handled (e.g., terminate processing) on this node. Files passing through the node destined for another node are not modified (e.g., if a file is passing through your node destined for another node, the default form field in the NJE file header is not modified).

---

# EBCDICTBL

## (Set Path for EBCDIC to ASCII Translation Table File)

### Purpose

The EBCDICTBL statement allows specification of an alternate EBCDIC to ASCII translation table.

### Format

<b>EBCDICTBL</b> _____ ▶—EBCDICTBL— <i>fullUnixpath</i> ————▶
--

### Parameters

#### **fullUnixpath**

The full Unix path specification and filename for the file to be used to store the EBCDIC/ASCII translation table. The default table is hardcoded in the application. Refer to “Character Set Translation” on page 92 for information on creating new translation tables.

### Usage

Typically changing this statement is only required if your country uses specific character glyphs in common practice. The default file implements the standard EBCDIC code page in use on z/VM, and is suitable for most users.

### Examples

An example of changing this parameter might appear similar to:

```
EBCDICTBL /etc/nje/e2a.tbl
```

### Comments

1. The default value supplied in the installation package is generally correct for most installations. If you need to change this parameter, please contact SNA support for additional assistance.

---

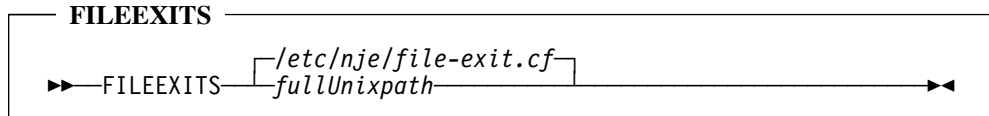
# FILEEXITS

## (Set Filename and Path for File Processing Exit Table)

### Purpose

The FILEEXITS statement specifies the pathname and file name containing the file processing exit table. Lines in this file specify what action is taken if a file arrives or departs according to a matching pattern in this file.

### Format



### Parameters

#### **/etc/nje/file-exit.cf**

Default location of file processing exit table. The default location is suitable for most users. If you need to change this parameter, please contact SNA support for assistance.

#### **fullUnixpath**

The full Unix path specification and filename for the file used to store the file processing exit table.

### Usage

Most users will not need to change this parameter. Please contact SNA support for additional assistance if you think you need to change this parameter.

### Examples

An example of this statement might appear similar to this:

```
FILEEXITS /etc/nje/file-exit.cf
```

### Comments

See “File Processing Exit (file-exit.cf)” on page 82 for more information on file processing exits and parameters.



---

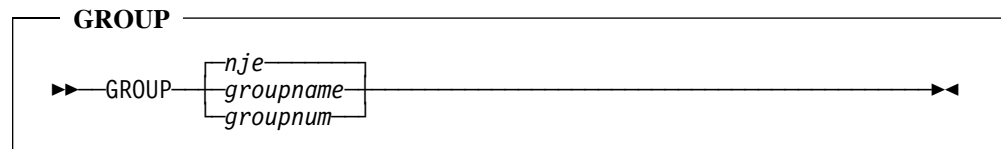
# GROUP

## (Set Unix Group Name for General NJE Users)

### Purpose

The GROUP statement indicates a Unix group that users of the NJE Bridge should belong to in order to use the NJE utilities.

### Format



### Parameters

#### nje

Default group name for general users. If group nje does not exist at installation, it is created by the install process.

#### groupname

Unix groupname that users of the NJE Bridge userspace tools should belong to. This entry must exist in /etc/groups. If you wish the utilities to be available to all users of the system, this value should be a group that all users of the system are a member of.

#### groupnum

A numeric Unix group number that users of the NJE Bridge userspace tools should belong to.

Note that while bare numeric entries work, good system administration practice should use group names in /etc/groups to avoid future conflicts with new groups, or having to update large numbers of userid entries in /etc/groups at a later date.

### Usage

You can restrict use of the NJE tools by placing them in a separate group and assigning users to that group on an individual basis. Otherwise, it is most convenient to use a common group such as users to allow all system users to use the NJE Bridge utilities. Check your distribution for the name used by your distribution for all users.

### Examples

An example of allowing all users of the system who are members of group users to use the NJE Bridge might appear similar to:

```
GROUP users
```

An example of specifying the group by number might appear similar to:

```
GROUP 1101
```

## Comments

1. Check `/etc/groups` on your system for group names and numbers in use on your system.

---

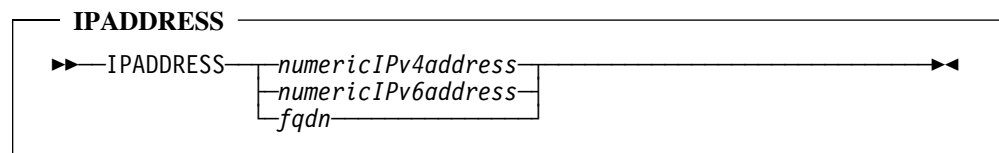
## IPADDRESS

### (Set IP Address Used in Link Signon)

#### Purpose

The IPADDRESS statement identifies the IP address used in link signon messages for this node.

#### Format



#### Parameters

##### **numericIPv4address**

Numeric IPv4 address for this host that should be used in NJE link signon messages. The address should be provided in dotted-quad format (e.g., A.B.C.D).

##### **numericIPv6address**

The preferred numeric IPv6 address for this host that should be used /Volumes/projects/43711-NJE-Bridge-Doc-Rewrite/Software-Referencein NJE link signon messages. The address can be provided in any valid IPv6 notation.

##### **fqdn**

Fully qualified domain name for this host. The name supplied here must have both forward and reverse entries in the DNS. For hosts with both IPv4 and IPv6 addresses, the code listens on the port specified by the IPPORT (see “IPPORT (Set IP Port for NJE Traffic On Remote Host for This Line)” on page 50) statement on both stacks, and the value returned by getaddrinfo is used for the IPv4 stack. The value specified on this statement is copied directly into the link signon record for IPv6 hosts as specified on this statement.

#### Usage

The IPADDRESS statement identifies the preferred IP address of this host for NJE traffic. Some NJE implementations use this address as part of the link signon validation process and require the address to match the one specified in the link definition in that implementation. VSE/POWER PNET is known to require the address in the signon record transmitted by a NJE node to match. Other implementations may choose to validate or ignore this value entirely.

#### Examples

An example of an IPADDRESS statement with a numeric IPv4 address might appear similar to this:

```
IPADDRESS 192.168.0.1
```

An example of an IPADDRESS statement with a numeric IPv6 address might appear similar to this:

```
IPADDRESS 0::0
```

If using a fully-qualified domain name (FQDN), the IPADDRESS statement might look similar to this:

```
IPADDRESS wizard.dwarf.example.com
```

## Comments

1. This statement is mandatory, and must be customized for your site.
2. The value of IPv4 address parameters in this statement is ignored by the z/VM implementation and other NJE Bridge nodes, but VSE/POWER implementations validate and respect the value listed here. For IPv6 hosts, the value of this parameter is not validated or used by any NJE implementation at this time.
3. For IPv4 stacks, if a DNS name is listed here, the address value in the first DNS A record for the DNS entry will be used as the value of the IP address parameter for this statement when contacted by a IPv4 host. For IPv6 stacks, if a DNS name is listed here, the address value in the first DNS AAAA record for the DNS entry will be used as the value of this parameter when contacted by a IPv6 host.

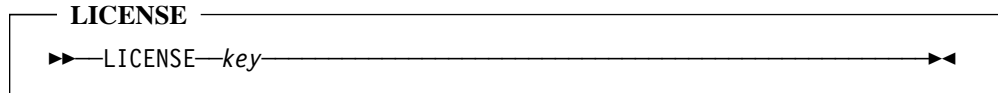
---

## LICENSE (Specify PAK for This Product)

### Purpose

The LICENSE statement identifies the product activation key for the NJE Bridge on this platform. The product activation key is required for the product to function. If you do not have a product activation key, contact SNA support for further information.

### Format



### Parameters

#### key

The product activation key (PAK) for this product. For the NJE Bridge, PAKs are 5 groups of letters and numbers separated by hyphens, e.g.:

XXXXX-XXXXX-XXXXX-XXXXX-XXXXX

### Usage

This statement is required for the product to operate. If you do not have a PAK, contact SNA support.

### Examples

An example of the LICENSE statement might appear similar to this (note that the example is not a valid PAK):

```
LICENSE 12345-ABCDE-6F7G8-H9KAL-BMCND
```

### Comments

1. The hyphens separating the 5 groups of letters and numbers are required.
2. Keys generated for different hardware architectures (x86\_64 and s390x, for example) are not interchangeable. You must use the correct key for your hardware architecture.

---

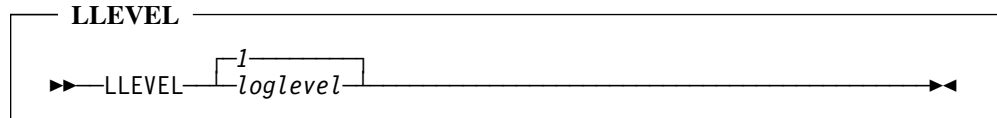
# LLEVEL

## (Set Default Log Detail Level)

### Purpose

The LLEVEL statement sets the default diagnostic logging detail level at NJE Bridge startup. The detail level may be adjusted up or down during operation using the ucp LOGLEVEL command.

### Format



### Parameters

1

Default log level. Normal operational detail logging is log level 1.

#### loglevel

Integer value from 1 to 6. As this value increases, the level of detail logged about internal operations within the NJE Bridge software increases.

On Unix and Linux systems, log data is written to the system log daemon (syslog).

Caution should be used with log levels higher than 4 for extended periods. Log levels higher than 4 dump extensive internal data about the internal protocol state machines and record structures and can quickly fill log files if used for extensive periods.

### Usage

This keyword allows debugging of the product internals. Please consult your support provider for instructions on how and when to use this statement. If you have problems with the product, please refer to “If You Have Trouble” on page 123.

### Examples

An example of setting a elevated level of logging detail might appear similar to this:

```
LLEVEL 6
```

Setting the log detail level to normal (1) would appear similar to this:

```
LLEVEL 1
```

### Comments

1. This parameter may have values higher than 6, but no additional detail is logged. Values greater than 6 are treated internally as 6.
2. In general, it is most valuable to leave the value of this statement in `nje.cf` at 1, and use the `ucp` command to dynamically modify this value during operations. By using this approach, the product will default to a reasonable level of log traffic at startup, minimizing the risk of filling up the filesystem containing the syslog data files.

---

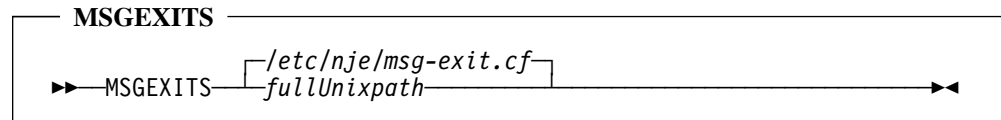
# MSGEXITS

## (Set Filename and Path for Message Processing Exit Table)

### Purpose

The MSGEXITS statement specifies the pathname and file name containing the message processing exit table. Lines in this file specify what action is taken if a NJE message arrives or departs according to a matching pattern in this file.

### Format



### Parameters

#### **/etc/nje/msg-exit.cf**

Default location of message processing exit table. This value is suitable for most users. If you need to change this parameter, please contact SNA support for additional assistance.

#### **fullUnixpath**

The full Unix path specification and filename for the file used to store the message processing exit table.

### Usage

Most users will not need to change this parameter. Please contact SNA support for additional assistance if you think you need to change this parameter.

### Examples

An example of this statement might appear similar to this:

```
MSGEXITS /etc/nje/msg-exit.cf
```

### Comments

See “Message Processing Exit (msg-exit.cf)” on page 86 for more information on file processing exits and parameters.

---

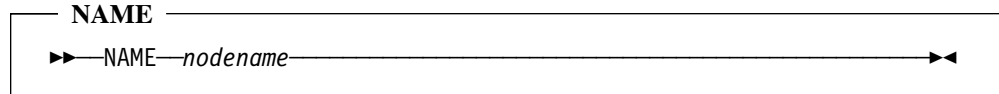
# NAME

## (Set Local NJE Node Name)

### Purpose

The NAME configuration statement sets the NJE node name of the this node.

### Format



### Parameters

#### **nodename**

A 1 to 8 character NJE node name.

### Usage

This statement should be the first non-comment statement in the `nje.cf` file.

### Examples

A sample of the NODE statement might appear as follows:

```
NODE NJENODE1
```

where NJENODE1 is the NJE node name of this system within your NJE network.

### Comments

1. This parameter is mandatory and must be a unique 8 character NJE node name in your network.
2. NJE node names may contain one (1) to eight (8) letters from A to Z, numbers from 0 to 9, and \$ (although use of special characters is discouraged as some other non-IBM NJE implementations do not permit special characters).



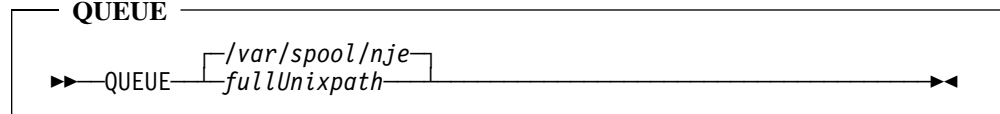
---

## QUEUE (Set Queue Directory Path)

### Purpose

The QUEUE statement sets the directory used as the spooling area for files sent and received by the NJE Bridge, including files queued to users on this node and files being processed as store-and-forward files being sent or received by this node destined for other nodes.

### Format



### Parameters

#### ***/var/spool/nje***

Default location of spool area.

#### ***fullUnixpath***

Fully specified Unix path for queued files and messages.

### Usage

This keyword allows the spool directory to be located on a device that has significant dedicated space for use as a spool area.

Note that the size of the filesystem where this directory resides is the limiting factor on the size of files that can be sent and received with the NJE Bridge product. This directory should be located on a device with sufficient space to hold the maximum amount of data to be sent or received by this node.

### Examples

The location of the spool directory can be changed using this parameter. If we wish to move the spool directory to a dedicated filesystem or device mounted as `/spool`, the statement would appear like this:

```
QUEUE /spool
```

### Comments

1. This parameter is mandatory, however unless you have significant cause to relocate the spool directory used, the default value specified in the `nje.cf` file supplied with the install package is generally acceptable and should be preserved.
2. The amount of space available in the specified filesystem and/or directory is the upper limit of the size of files that can be transmitted or received by the NJE Bridge software. If this directory or device fills up, file and message transfer will halt and connected links will begin queuing files until space is available.
3. If your NJE node is very active and the number of files transmitted and received is large, you may need to use `fstune` to allocate more inodes to the filesystem used to hold this directory. If possible, you should use an advanced filesystem such as `xfs`, `gluster` or `ceph` for spool directories on very active systems.

---

## ROUTE\_TABLE

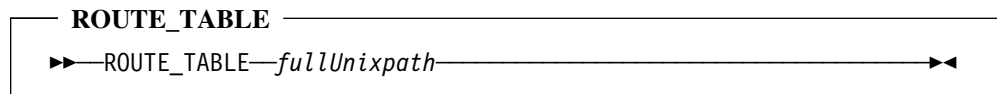
### (Set Filename and Path for NJE Routing Table - Alt Syntax)

#### Purpose

The ROUTE\_TABLE statement is provided for backward compatibility with previous versions of the NJE Bridge. It provides the same function as the TABLE statement. New implementations should use the TABLE statement and older implementations should update `nje.cf` to use the TABLE statement.

Please refer to “TABLE (Set Filename and Path for NJE Routing Table)” on page 43 for information on the TABLE statement.

#### Format



#### Parameters

##### fullUnixpath

The full Unix path specification and filename for the file to be used to store the NJE routing table. This file is built from the line definitions in `nje.cf` and explicit routes added by the ROUTE command in `ucp`. The default for this value is `/var/run/nje.route` and should not be changed without compelling reason.

#### Usage

This statement is deprecated. Use the TABLE statement in place of this statement.

#### Examples

This statement is deprecated. Use the TABLE statement in place of this statement.

#### Comments

1. Use of this statement is deprecated and will be removed in a future release. See “TABLE (Set Filename and Path for NJE Routing Table)” on page 43 for conversion information to the TABLE statement.

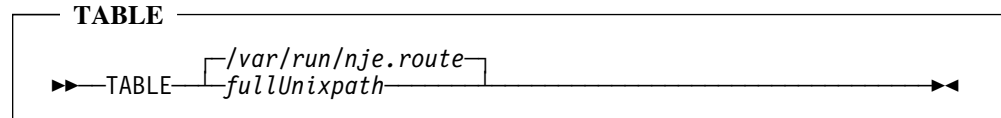
---

## TABLE (Set Filename and Path for NJE Routing Table)

### Purpose

The TABLE statement identifies the Unix path and filename of the file used to store the NJE routing table. The file is stored in binary, and is sensitive to machine word endianness.

### Format



### Parameters

#### `/var/run/nje.route`

Default location of NJE routing table. The default value is suitable for most users. Contact SNA support if you think you need to change this value.

#### `fullUnixpath`

The full Unix path specification and filename for the file to be used to store the NJE routing table. This file is built from the line definitions in `nje.cf` and explicit routes added by the ROUTE command in `ucp`.

### Usage

Normally there is no reason to change this value, but if the NJE implementation is located in a cluster of systems, it may be necessary to change this value to locate it on a shared disk volume. If you need to implement a clustered environment, please open a support ticket to discuss the implications of such a configuration.

### Examples

An example of this statement might appear similar to:

```
TABLE /shared/volume/nje.route
```

### Comments

1. This statement is mandatory, and should be left at the default value unless recommended by SNA support.

---

# USEREXITS

## (Set Filename and Path for File Processing Exit Table - Alt Syntax)

### Purpose

The USEREXITS statement is provided for backward compatibility with previous versions of the NJE Bridge. It provides the same function as the FILEEXITS statement. New implementations should use the FILEEXITS statement and older implementations should update nje.cf to use the FILEEXITS statement.

Please refer to “FILEEXITS (Set Filename and Path for File Processing Exit Table)” on page 32 for information on the FILEEXITS statement.

### Format

<b>USEREXITS</b> ————— ▶—USEREXITS— <i>fullUnixpath</i> —————▶
---

### Parameters

#### **fullUnixpath**

The full Unix path specification and filename for the file to be used to store the file processing table.

### Usage

This statement is deprecated. Use the FILEEXITS statement in place of this statement.

### Examples

This statement is deprecated. Use the FILEEXITS statement in place of this statement.

### Comments

1. Use of this statement is deprecated and will be removed in a future release. See “FILEEXITS (Set Filename and Path for File Processing Exit Table)” on page 32 for conversion information to the FILEEXITS statement.

---

## Line-Specific Configuration Statements (nje.cf)

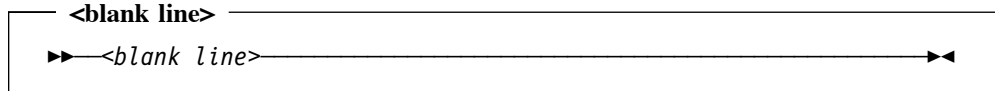
The following statements affect individual NJE links or "lines" and are used to specify parameters used for connections to a specific NJE host. Some of these parameters must match the definitions used on the remote end of the connection. If the parameter must match, a note in the comments for the statement will indicate that the parameter must match the value specified on the remote end of the link. The definition of a line is terminated by a blank line in the nje.cf file.

---

## <blank line> (End Statement Block)

### Purpose

### Format



### Parameters

#### <blank line>

A blank line terminates the current statement block.

Note that the statement is NOT visible - it is a empty line.

### Usage

Insert a blank line following each group of statements. If used at the end of a line definition, the blank line terminates the definition of the line.

### Examples

An example of the use of a blank line might appear similar to:

```
<blank line>  
LINE 10 SNARF1  
.  
. (other line definition statements)  
.  
<blank line>  
LINE 11 SNAFU  
.  
. (other line definition statements)  
.  
<blank line>
```

### Comments

1. On all platforms, a blank line is constituted of no characters followed by the platform line-end sequence (LF on Unix, default on other platforms).

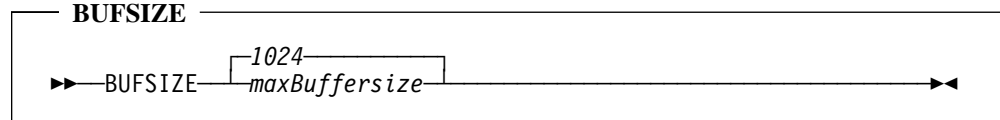
---

# BUFSIZE

## (Set Maximum Transfer Buffer Size for This Line)

### Purpose

### Format



### Parameters

#### 1024

Default buffer size if not otherwise specified.

#### maxBufferSize

Maximum transmit and receive buffersize used for this link. This value represents the maximum size accepted by this link; if the remote host uses a smaller size, the two nodes negotiate down to the smaller size.

### Usage

This statement should reflect a value evenly divisible by the maximum data payload of individual packets between the two hosts. Odd values here will force unnecessary fragmentation of packets and correspondingly, will perform less efficiently.

### Examples

If a remote host has declared a buffer size of 4096 for an IPv4 connection, the BUFSIZE statement might appear similar to:

```
LINE 9 SNABAR
TYPE TCPV4
BUFSIZE 4096
.
.
.
```

### Comments

1. This parameter must be greater the value specified on the remote side. The line will not start correctly if the value is different than the remote specified value. The IBM implementations are particularly sensitive to this value being different; other NJE Bridge nodes will negotiate this value more gracefully.

---

# CERTIFICATE

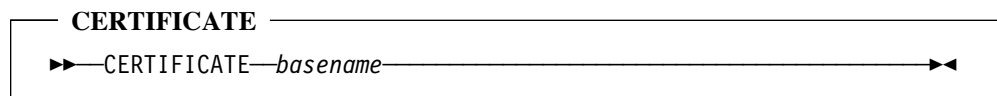
## (Set Certificate File Base Name to Use for This Line)

### Purpose

The CERTIFICATE statement provides the base file name of the certificate file to be used for this connection. The filename will have a extension of `.crt` appended to it, and the directory specified by the CERTDIR general configuration statement (see “CERTDIR (Set Path for SSL-related Information)” on page 26 for more information on the general configuration statement).

This statement is deprecated and will be removed in a future release.

### Format



### Parameters

#### basename

The base portion of the filename containing the certificate. The filename will have a `.crt` extension added and the directory specified in the CERTDIR general configuration statement will be searched for the resulting filename.

### Usage

This statement applies only to line types TCPV4SSL and TCPV6SSL.

### Examples

A line definition using a certificate might appear similar to:

```
LINE 9 SNABAR
TYPE TCPV4SSL
TCPNAME wizard.penguin.example.com
CERTIFICATE wizard
.
.
.
```

### Comments

1. This statement is deprecated and is included here only for documentation purposes. Replace this statement with CERTFILE.
2. This statement is optional unless SSL encryption is desired (line types TCPV4SSL and TCPV6SSL). If specified for a non-SSL link, the statement is ignored.
3. A system-wide certificate called `njeCA.crt` containing certification authority chains for the complete certificate tree must be installed if not using the supplied self-signed certificate.
4. Certificates can be in any format supported by OpenSSL. The file name is not sensitive to the certificate format.

See “SSL-Related Files” on page 71 for more information on using SSL to encrypt data transfer.



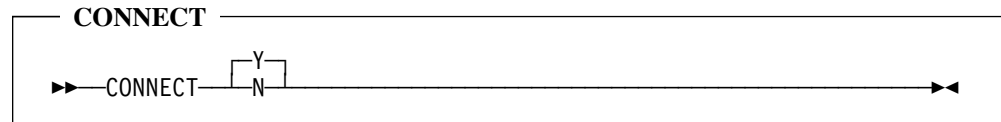
---

## CONNECT (Control Connection Initiation for This Line)

### Purpose

The CONNECT statement controls whether the NJE Bridge code will actively initiate connections to the remote host for this line, or wait passively for an incoming connection from the remote host.

### Format



### Parameters

**Y**

The NJE Bridge will attempt active connections to the specified host.

**N**

The NJE Bridge will not attempt active connections to the specified host. The node will listen for incoming connections.

### Usage

This keyword is used in cases where firewall or company policy does not permit active connections to outside hosts, or the firewall policy may be such that active connections cannot resolve at system startup.

### Examples

The following example sets up link SNABAR to listen for incoming connections from the remote host:

```
LINE 9 SNABAR
TYPE TCPV4
CONNECT N
```

### Comments

1. If NJE is running behind a firewall, often there is a policy that forbids the remote end connecting to the peer. In this case NJE will wait on a connect() call until it times out.

---

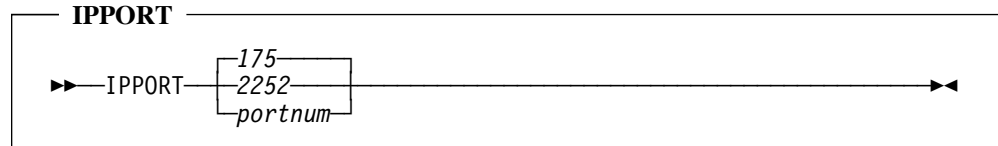
# IPPORT

## (Set IP Port for NJE Traffic On Remote Host for This Line)

### Purpose

The IPPORT statement specifies the TCP port on the remote system to use when connecting to that node. This statement is relevant only when this node is connecting to the remote system; connections to this node are assumed to use TCP port 175.

### Format



### Parameters

#### 175

Default TCP port number registered with IANA for NJE over TCP.

#### 2252

Default TCP port number registered with IANA for SSL-protected NJE over TCP.

#### portnum

TCP port number to use when connecting to the remote NJE host. The same port number is used for both IPv4 and IPv6 connections.

### Usage

The default value of TCP port 175 (for non-SSL connections) and 2252 (for SSL connections) are normally suitable for most uses. This statement is most frequently needed in situations where firewalls are used and a different port on the firewall host is needed (frequently a firewall administrator will open a port on the firewall and redirect it to an internal host, hiding the internal network details from an outside system. To do this, the firewall administrator will use a unique TCP port number and forward that to port 175 on the NJE host).

### Examples

An example might look like this:

```
LINE 9 SNABAR
TYPE IPV4
IPPORT 2099
BLKSIZE 4096
.
.
.
```

### Comments

1. Note that if the firewall setup mentioned above is used, you may also need NATIN and NATOUT statements if the firewall also uses network address translation (NAT), especially if multiple levels of NAT are applied by multiple firewalls.
2. The specified port number is used for both IPv4 and IPv6 connections.

---

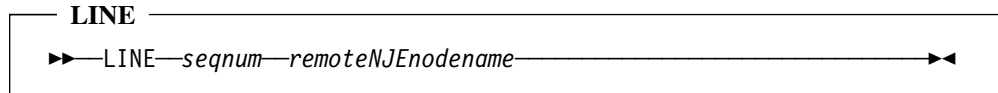
# LINE

## (Define A Connection to Another NJE Host)

### Purpose

The LINE statement defines a connection to another NJE host. Each connection has a number of parameters describing the connection to the NJE host; some must match the same parameter defined on the remote system.

### Format



### Parameters

#### seqnum

A integer from 1 to 32786 uniquely identifying this connection. The numbers used do not have to be sequential, but must be unique within this NJE node.

#### remoteNJEhostname

The NJE node name of the system on the remote end of this connection. This parameter must match the primary NJE node name defined on the remote system, and must be a valid NJE node name.

### Usage

The LINE statement is used for all connections to other NJE hosts. There must be a blank line (CR/LF only) before each LINE statement and accompanying parameter statements, and a similar blank line ending the definitions for a specific LINE.

### Examples

An example of a LINE definition might appear similar to this:

```
<blank line>
LINE 1 SNABAR
TYPE TCPV4
.
.
.
<blank line>
LINE 2 SNAF2
.
.
.
```

### Comments

1. At least one LINE statement and parameter entries must exist in nje.cf in order for the product to start correctly. The product includes a number of sample entries that may be removed once product verification is complete and at least one local LINE entry has been defined.

---

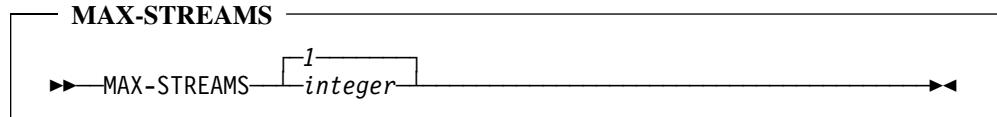
# MAX-STREAMS

## (Set Maximum Number of Parallel Transfer Streams for This Line)

### Purpose

The MAX-STREAMS statement specifies the number of parallel transfer streams (in- and out-bound) that are permitted on this connection.

### Format



### Parameters

1

Default number of streams for this line.

#### **integer**

A numeric value from 1 to 7. The value specified here must match the entry on the remote end or the line will stop when the remote system attempts to activate more streams than are defined here. 7 streams is the maximum allowed by the NJE protocol.

### Usage

In most cases, the more streams defined, the more quickly files can be transferred between nodes. The IBM default of 1 stream is conservative, and the allocation of files between streams is implementation-dependent (see “ORDER (Set Queuing Mechanism and Transmission Sequencing for This Line)” on page 55 for how files are assigned to streams in the NJE Bridge implementation).

### Examples

An example of the MAX-STREAMS statement might look like this:

```
MAX-STREAMS 7
```

### Comments

1. As noted above, more streams is generally better. If the value specified here does not match the entry on the remote system, the link will appear to operate normally for low traffic volumes, but will begin to fail randomly as traffic ramps up.
2. Most IBM implementations use the OPARM keyword to assign files to streams based on size. Consult the documentation for your NJE implementation to determine how to specify stream ids and file sizes on your operating system.
3. The ORDER keyword allows other nodes implemented by NJE Bridge software to allow prioritization of files based on other criteria. Refer to “ORDER (Set Queuing Mechanism and Transmission Sequencing for This Line)” on page 55 for more information.

---

# NATIN

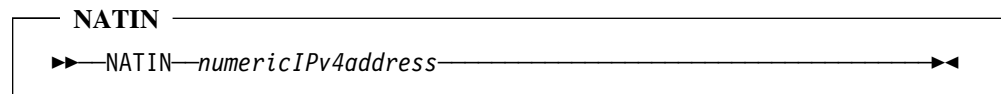
## (Specify "Inside" Address Value for NAT Firewall Support)

### Purpose

The NATIN statement specifies an IPv4 address to be used when constructing "OPEN" requests. This allows NJE to communicate with a peer on the other side of a NAT-based firewall.

This address specifies the "behind-the-firewall" IPv4 address of the remote host, to be used in matching the address of the system to the IP address in the NJE packet headers when used inside a firewall that performs NAT on IPv4 traffic.

### Format



### Parameters

#### **numericIPv4address**

The numeric IPv4 address of the remote system as seen inside the firewall in dotted-quad format, e.g., A.B.C.D.

### Usage

This parameter applies only to IPv4 traffic, which includes the IP address of the destination system inside the NJE connection protocol headers. NJE over IPv6 does not insert addresses inside the protocol headers, and thus does not need this parameter.

This keyword is normally only required in the case of multiple NAT firewalls separating the two hosts. Contact SNA support if you think you need to use this keyword.

### Examples

A NATIN statement might look like this:

```
NATIN 192.168.0.1
```

### Comments

1. The address specifies the "behind-the-firewall" IPv4 address of the remote host. It allows specification of the actual internal IPv4 address to be used in matching the address of the system to the IP address in the NJE packet headers when used inside a firewall that performs NAT on IPv4 traffic.
2. Use of the TNPASS, TLPASS, RNPASS and RLPASS statements may avoid the complexity of using this statement. See the respective entries for those commands for more details.

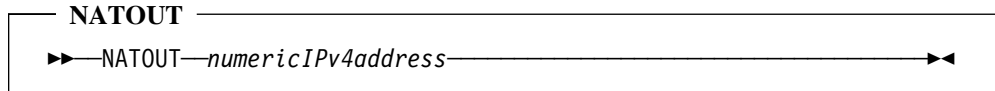
---

# NATOUT

## (Specify "Outside" Address Value for NAT Firewall Support)

### Purpose

### Format



### Parameters

#### **numericIPv4address**

the numeric IPv4 address of the remote system as it is to be seen by the remote host outside the NAT firewall.

### Usage

This parameter applies only to IPv4 traffic, which includes the IP address of the destination system inside the NJE connection protocol headers. NJE over IPv6 does not insert addresses inside the protocol headers, and thus does not need this parameter.

This keyword is normally only required in the case of multiple NAT firewalls separating the two hosts. Contact SNA support if you think you need to use this keyword.

### Examples

A NATOUT statement might look like this:

```
NATOUT 224.100.0.1
```

### Comments

1. The address specifies the "outside-the-firewall" IPv4 address of the remote host. It allows specification of the actual external IPv4 address to be used in matching the address of the system to the IP address in the NJE packet headers when used with a firewall that performs NAT on IPv4 traffic. VSE systems are particularly sensitive to this parameter setting.
2. Use of the TNPASS, TLPASS, RNPASS and RLPASS statements may avoid the complexity of using this statement. See the respective entries for those commands for more details.

---

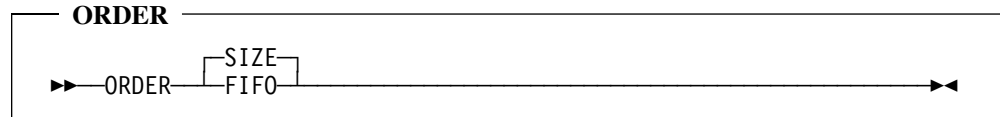
# ORDER

## (Set Queuing Mechanism and Transmission Sequencing for This Line)

### Purpose

The ORDER statement allows you to prioritize files for transmission by size or by arrival time (essentially guaranteeing first-in, first-out semantics (FIFO)).

### Format



### Parameters

#### SIZE

Files are assigned to streams based on size. This is the default setting.

#### FIFO

Files are assigned to streams based on arrival time, e.g. the first file to arrive is first to be assigned on an outbound stream.

### Usage

Use this statement only when the order of the files arriving at the remote system must be maintained. Use of this parameter may slow down file arrival if a strict FIFO queue must be maintained.

### Examples

Using the ORDER statement to enforce file sequencing might look similar to:

```
ORDER FIFO
```

### Comments

1. Note that the use of ORDER FIFO for links with multiple streams effectively delivers a single stream due to the enforcement of arrival times. Multiple files may arrive and be scheduled for transmission on multiple streams, but streams are used one at a time to ensure arrival order is maintained. The remote system is not aware of this processing; it simply sees files sent in the correct order with streams made selectively idle when waiting for their turn at transmission.

---

# PATHMGR

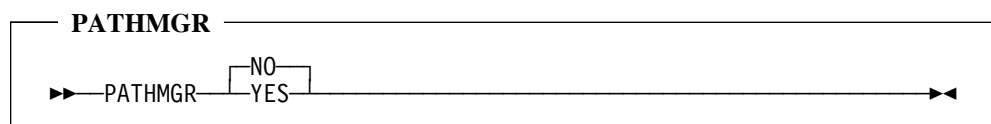
## (Enable JES2 PATHMGR Dynamic Routing Record Support)

### Purpose

The PATHMGR statement enables or disables support for the JES2 dynamic topology and routing support in the NJE protocol. At this time, only the NJE Bridge and the IBM JES2 on z/OS implementation have support for this feature.

If PATHMGR is set to YES, this link accepts and processes NJE M and N records from other PATHMGR-enabled nodes, building a dynamic network topology tree and inserting appropriate ROUTE entries to match the topology tree.

### Format



### Parameters

#### YES

This link and node can process type N and M NJE network topology and routing update messages. This link also receives topology change information for lines directly connected to this node (regardless of path manager support for the link).

#### NO

This link and node does not understand NJE network topology and routing update messages, and does not send topology and routing update messages. The up/down state of this link does generate network topology and routing updates for all other PATHMGR YES links attached to this node.

### Usage

For compatibility with the IBM implementations, the default is PATHMGR NO. To specify that this line should receive and generate topology and routing update records (i.e., the remote system is either JES2 or another NJE Bridge system), include the PATHMGR YES statement in the line definition.

### Examples

An example line definition with PATHMGR support enabled might appear similar to:

```
LINE 9 SNABAR
.
.
PATHMGR YES
.
.
```

### Comments

1. IBM implementations on systems other than JES2 will halt the link if PATHMGR records are sent to a node that does not implement path manager functionality. For JES3, RSCS, iSeries and VSE links, set PATHMGR to NO.
2. For the IBM JES2 implementation, dynamic topology updates are sent only for nodes with PATHMGR=Y specified, i.e., if a JES2 node has two links, one with



PATHMGR=Y and the other with PATHMGR=N, and the link with PATHMGR=N changes state to down, the JES2 implementation will not send topology updates to PATHMGR=Y nodes about the state change.

The NJE Bridge implementation generates updates for any connected link state change to any link with PATHMGR YES specified, e.g. in our example above, the state change of the PATHMGR=N link will generate a topology update to any connected links with PATHMGR=Y.

3. If using PATHMGR extensively, use of the RESISTANCE statement on all line definitions is highly recommended. This allows the path manager code to prioritize the routes used by normal traffic appropriately, while still maintaining a full topology map of all connections.
4. If using a combination of systems that can and cannot support PATHMGR YES definitions, it may be worth considering making downstream nodes with PATHMGR NO spokes connecting directly to a NJE Bridge node to take advantage of the ability of the NJE Bridge code to reflect topology changes to non-PATHMGR links to other PATHMGR nodes. In general, if your other network nodes can handle PATHMGR YES, it greatly simplifies network operations in that no local routing tables need be maintained on JES2 and NJE Bridge nodes.

---

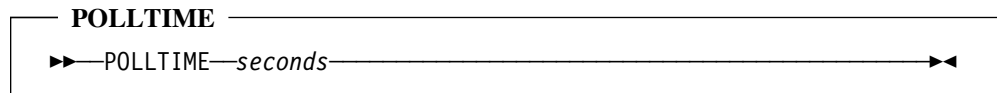
# POLLTIME

## (Specify Wait Time When Expecting Incoming Connection Request)

### Purpose

The POLLTIME statement inserts a small random salt in the amount of time a node waits for a connection from a remote host. This random delay is used to ensure the two nodes do not deadlock waiting for the other side to connect.

### Format



### Parameters

#### **seconds**

This parameter specifies a time from 30 seconds to  $2^{31}-1$  seconds. The default value is 43 seconds plus/minus a random value between 0 and 15 seconds (e.g., a random value between 28 and 58 seconds).

### Usage

This parameter is typically only required on asymmetric or unreliable links where there may be a delay in the remote system responding to a signon sequence. Contact SNA support if you think you need to use this statement.

### Examples

An example of the POLLTIME statement might look similar to this:

```
POLLTIME 62
```

### Comments

1. This statement is rarely required. Contact SNA support if you think you need to use this statement.

---

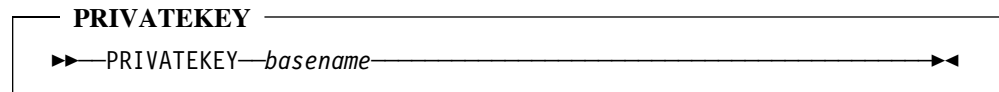
# PRIVATEKEY

## (Set Private Key File Base Name to Use for This Line)

### Purpose

The PRIVATEKEY statement provides the base file name for the private key to be used to decrypt the certificate specified by the CERTIFICATE statement for this line (see “CERTIFICATE (Set Certificate File Base Name to Use for This Line)” on page 48 for usage notes).

### Format



### Parameters

#### **basename**

The base portion of the filename containing the private key for the certificate specified by the CERTIFICATE statement for this line.

### Usage

If your certificates use private keys, you must supply a PRIVATEKEY statement to indicate the key to be used to decrypt the certificate. The extension `.key` is appended to the `basename` and the application searches the directory specified in the CERTDIR general configuration statement (see “CERTDIR (Set Path for SSL-related Information)” on page 26 for details of the general configuration statement) for the resulting file.

### Examples

An example of use of the PRIVATEKEY statement might appear similar to: The certificate for the line is in file `snabar.crt` and the private key for that certificate is in file `snabar.key`.

```
LINE 9 SNABAR
TYPE TCPV4SSL
CERTIFICATE snabar
PRIVATEKEY snabar
.
.
.
```

### Comments

1. The syntax of this statement is the same for both IPv4 and IPv6 links.
2. The private key can be in any format supported by OpenSSL. The file name and the key representation are not interlinked. The filename should have an extension of `.key` regardless of key format.

See “SSL-Related Files” on page 71 for more information on using SSL encryption.

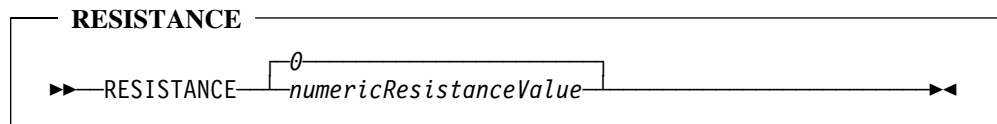
# RESISTANCE

## (Specify Line Resistance Value for PATHMGR Dynamic Routing)

### Purpose

The RESISTANCE statement provides a way to specify weighting of this link when choosing transmission paths between nodes. When PATHMGR is active, the outbound link selection algorithm chooses the link with the lowest resistance value in the routing table built by the dynamic topology build process.

### Format



### Parameters

**0**

Default resistance value.

#### **numericResistanceValue**

A numeric integer from 0 to 65535 indicating a weighting to be placed on use of this connection.

### Usage

The value of this parameter is used to calculate preference for link usage when PATHMGR is active. Lower values indicate a link is preferred vs other links between the same links. The default value is 0.

### Examples

One example might be the case where a pair of nodes have both IPv4 and IPv6 connectivity, but the IPv6 connection is lower bandwidth, and thus we want to prefer the IPv4 connection unless there is no other option. Specifying a higher resistance value might appear similar to the following:

```

LINE 9 SNABAR
TYPE TCPV4
.
.
RESISTANCE 100
.
.
<blank line>
LINE 10 SNABAR6
TYPE TCPV6
RESISTANCE 1000
.
.
.
  
```

The higher resistance value for the TCPV6 link will cause the majority of the NJE traffic to flow over the IPv4 connection, but will automatically switch to the IPv6 connection if the IPv4 link fails.

## Comments

1. Note that this value has no effect on links that do not support PATHMGR YES. All PATHMGR NO links are automatically assigned a resistance of 0.
2. If this statement is added to a PATHMGR NO link, it is parsed but the value supplied is ignored and a resistance of 0 is used.
3. This parameter is not required, but is highly recommended if you anticipate using PATHMGR in the future, or there are multiple connections between two nodes (such as the case between two nodes with both IPv4 and IPv6 connections).

---

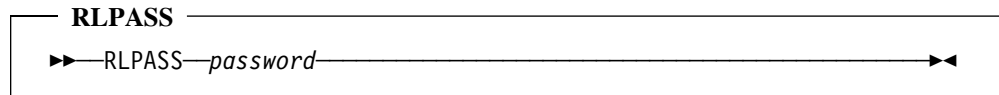
# RLPASS

## (Set Line Password Expected by Remote Line)

### Purpose

If specified, this is the line password the peer node is expecting for node's line. This will be checked at link signon time.

### Format



### Parameters

#### password

1 to 8 character line/link password. This value must match the line password value expected for this line by the remote system.

### Usage

This parameter is implemented for communicating with JES2 and JES3 NJE implementations where multiple connections can be treated as a single logical connection. The NJE Bridge does not currently implement this capability, but defines the keywords to support it for a future release, and validates the password supplied if the statement is used.

### Examples

An example of this statement might appear similar to this:

```
RLPASS $RR1I0$$
```

### Comments

1. This implementation of NJE has a one-to-one correspondence between nodes and lines but JES2 and JES3, for example, do not. If the remote system supports both node and line passwords and both node and line password statements are used, the password for node and line on the remote system must match the entry supplied by this system.

---

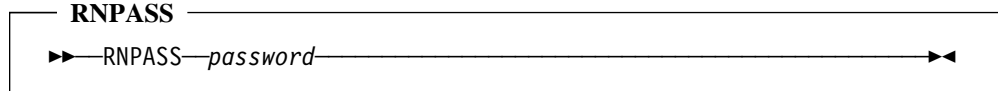
# RNPASS

## (Set Node Password Expected by Remote Node)

### Purpose

The RNPASS statement specifies the node password expected by the remote system for this connection. Validation and acceptance of this password is determined by the remote system.

### Format



### Parameters

#### password

1 to 8 character link password. This value must match the value expected by the remote system.

### Usage

### Examples

We wish to define a connection from local node SNAFU to remote node SNABAR. On remote node SNABAR, a link definition for our local node SNAFU is defined with a transmit node password of `$$$UOT$$$` and a received node password of `$100$18`. The line definition on node SNAFU would appear similar to:

```
LINE 9 SNABAR
TYPE TCPV4
TNPASS $$$UOT$$$
RNPASS $100$18
.
.
.
```

### Comments

1. In most cases, the use of node passwords is easier to maintain than using the NATIN and NATOUT keywords. If communicating with a VSE/POWER node via IPv4, you must use NATIN and NATOUT as VSE verifies IP addresses in packets. For all other NJE implementations, TNPASS and RNPASS can be used in place of NATIN/NATOUT.
2. Note that line and node passwords are passed in clear text, but do not imply or allow any login access to the system in question. If the remote system supports SSL, it is wise to use SSL protection for the link, but not required. Only the JES2 and JES3 IBM implementations implement SSL-protected links; other implementations should front-end their outside connections with a copy of the NJE Bridge (with SSL enabled) to prevent disclosing the node password.
3. Acceptance of a node password is determined by the system authenticating the password. If the remote system requires additional authentication (for example, VSE requiring IP addresses to match), then the local node must supply sufficient information to satisfy the requirement of the remote system.

---

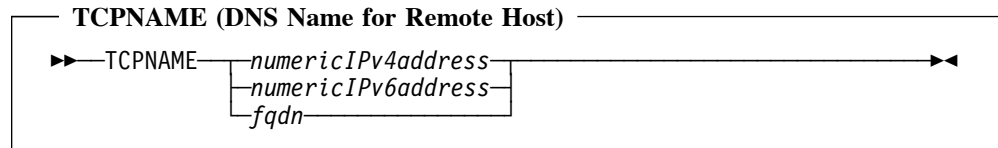
# TCPNAME

## (Set DNS Name for Remote Host for This Line)

### Purpose

The TCPNAME statement provides the DNS hostname or IP address of the remote host. The combination of the hostname and the value of the IPPORT statement are used to initiate connections from this host to the remote system.

### Format



### Parameters

#### **numericIPv4address**

Numeric IPv4 address of the remote host in dotted-quad format, e.g., A.B.C.D.

#### **numericIPv6address**

Numeric IPv6 address of the remote host in any valid IPv6 format.

#### **fqdn**

Fully qualified domain name of the remote host. If the line is of type TCPV4 or TCPV4SSL, the address provided from the first A record in the response from the DNS server is used. If the line is of type TCPV6 or TCPV6SSL, the address provided from the first AAAA record in the response from the DNS server is used.

### Usage

As noted above, this is the IP address of the remote host to be used for connections. An address is required for each link even if only incoming connections are used.

### Examples

An example of a numeric IPv4 address connection (with and without SSL) might appear similar to:

```
TCPNAME 10.10.10.10
```

An example of a numeric IPv6 address connection (with and without SSL) might appear similar to:

```
TCPNAME 0::0
```

An example of a connection using a FQDN can look like this:

```
TCPNAME wizard.penguin.example.com
```

### Comments

1. This parameter is required.
2. Use of numeric IPv4 and IPv6 addresses should be discouraged. Hosts move around, get renumbered, and hard-coding IP addresses in applications is generally a bad idea. Use of a FQDN also allows the same name to be used and have the name resolve



properly for both IPv4 and IPv6 connections without the confusion of different address statements and formats in application entries.

---

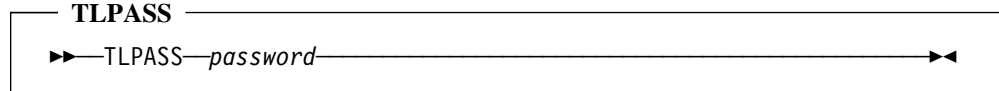
# TLPASS

## (Set Line Password Expected From Remote Line)

### Purpose

If specified, this is the password we are expecting from the peer node's line. This will be checked at link signon time. This implementation of NJE has a 1:1 correspondence between nodes and lines but JES2, for example, does not.

### Format



### Parameters

#### password

1 to 8 character link password. This value must match the line password value supplied by the remote system.

### Usage

This parameter is implemented for communicating with JES2 and JES3 NJE implementations where multiple connections can be treated as a single logical connection. The NJE Bridge does not currently implement this capability, but defines the keywords to support it for a future release, and validates the password supplied if the statement is used.

### Examples

An example of this statement might appear similar to this:

```
TLPASS $TR1I0$$
```

### Comments

1. This implementation of NJE has a one-to-one correspondence between nodes and lines but JES2 and JES3, for example, do not. If the remote system supports both node and line passwords and both node and line password statements are used, the password for node and line on this system must match the entry supplied by the remote system.

---

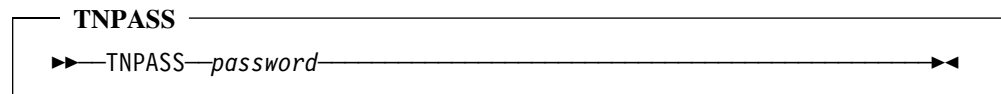
# TNPASS

## (Set Node Password Expected From Remote Node)

### Purpose

If specified, this value is the transmit node password we are expecting from the peer node. This will be checked at link signon time.

### Format



### Parameters

#### password

1 to 8 character link password. This value must match the value supplied by the remote system.

### Usage

This statement can be used to supply additional authentication information in situations where IPv4 address information can be unreliable (such as when multiple NAT firewalls are present). The NJE Bridge NJE implementation checks incoming connections to determine if

1. the connection matches an IP address specified in the TCPNAME parameter of a link
2. If no IP address matches any line definition, the connection is allowed to send a NJE OPEN record with a nodename and TNPASS value. If the nodename and TNPASS match, the link signon is permitted to proceed.
3. If the TNPASS value supplied by the remote node does not match the TNPASS value specified in `nje.cf` for this line, the connection attempt is rejected.

### Examples

We wish to define a connection from local node SNAFU to remote node SNABAR. On remote node SNABAR, a link definition for our local node SNAFU is defined with a transmit node password of `$$$UOT$$`. The line definition on node SNAFU would appear similar to:

```
LINE 9 SNABAR
TYPE TCPV4
TNPASS $$$UOT$$
.
.
.
```

### Comments

1. In most cases, the use of node passwords is easier to maintain than using the NATIN and NATOUT keywords. If communicating with a VSE/POWER node via IPv4, you must use NATIN and NATOUT as VSE verifies IP addresses in packets. For all other NJE implementations, TNPASS and RNPASS can be used in place of NATIN/NATOUT.
2. Note that line and node passwords are passed in clear text, but do not imply or allow any login access to the system in question. If the remote system supports SSL, it is

wise to use SSL protection for the link, but not required. Only the JES2 and JES3 IBM implementations implement SSL-protected links; other implementations should front-end their outside connections with a copy of the NJE Bridge to prevent disclosing the node password.

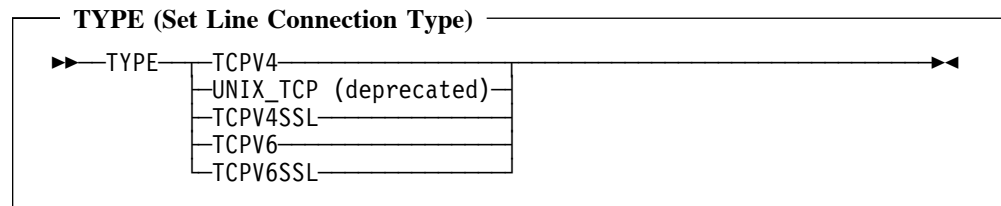
---

# TYPE

## (Set Type of Connection Method for This Line)

### Purpose

### Format



### Parameters

#### TCPV4

IPv4 connection without SSL encryption.

#### UNIX\_TCP

Synonym for TCPV4 for backward compatibility. This parameter is deprecated and will be removed in a future release. If you are using this parameter, please replace it with TCPV4.

#### TCPV4SSL

IPv4 connection with SSL encryption. See “SSL-Related Files” on page 71 for more information about SSL-protected connections.

#### TCPV6

IPv6 connection without SSL encryption.

#### TCPV6SSL

IPv6 connection with SSL encryption. See “SSL-Related Files” on page 71 for more information about SSL-protected connections.

### Usage

The TYPE keyword indicates that the connection should be made using the specified type.

### Examples

A connection to node SNABAR using IPv4 with no encryption might look like this:

```
<blank line>
LINE 9 SNABAR
TYPE TCPV4
.
.
.
```

A IPv4 connection with SSL encryption might appear similar to:

```
<blank line>
LINE 9 SNABAR
TYPE TCPV4SSL
CERTIFICATE snabar
PRIVATEKEY snabar
```

```
.
.
.
```

Note that the TCPV4SSL type requires specification of the CERTIFICATE statement (and optionally the PRIVATEKEY statement if private keys are used). See “SSL-Related Files” on page 71 for more information about SSL certificates and keys.

A IPv6 connection might appear similar to:

```
<blank line>
LINE 9 SNABAR
TYPE TCPV6
```

```
.
.
.
```

And finally, a IPv6 connection with SSL encryption would appear as:

```
<blank line>
LINE 9 SNABAR
TYPE TCPV6SSL
CERTIFICATE snabar
PRIVATEKEY snabar
```

```
.
.
.
```

As noted for TCPV4SSL links, TCPV6SSL links require certificate and private key files (if private keys are used). See “SSL-Related Files” on page 71 for more details on SSL certificates and keys.

## Comments

1. This parameter is mandatory and must agree with the definition at the remote end of the link.
2. If you wish to have both IPv4 and IPv6 connections to the same NJE host, you will need to create two links with different names and specify a type of one of the IPv4 choices for one link, and one of the IPv6 choices for the other link.
3. Multiple links from this host to another host should explicitly specify resistance values if you plan to use PATHMGR support with JES2 or other NJE Bridge nodes. The link with the lowest resistance value will be selected if available. If equal resistance values are specified (e.g., all links from node A to node B have the same resistance value), the link used from A to B is chosen first according to the link with available transmission resources, and then in alphabetical sort order.

---

## SSL-Related Files

The NJE Bridge implements SSL-encrypted lines to IBM JES2/JES3 and other NJE implementations. Other IBM NJE implementations do not implement SSL encryption; however the NJE Bridge can front-end non-z/OS implementations, allowing encrypted connectivity with systems that do not natively implement SSL. This section discusses the files and statements necessary to implement a SSL-encrypted link between two NJE nodes.

---

## SSL Basics

SSL (Secure Sockets Layer) is a standard security technology for establishing an encrypted link between a server and a client - for example, a web server (website) and a browser; or a mail server and a mail client (e.g., Outlook).

SSL allows sensitive information such as credit card numbers, social security numbers, and login credentials to be transmitted securely. Normally, data sent between clients and servers is sent in plain text - leaving you vulnerable to eavesdropping. If an attacker is able to intercept all data being sent between a client and a server, then he or she can see and use that information.

More specifically, SSL is a security protocol. Protocols describe how algorithms should be used; in this case, the SSL protocol determines variables of the encryption for both the link and the data being transmitted.

SSL secures millions of peoples' data on the Internet every day, especially during online transactions or when transmitting confidential information. Internet users have come to associate their online security with the lock icon that comes with an SSL-secured website or green address bar that comes with an extended validation SSL-secured website. SSL-secured websites also begin with https rather than http.

### Where Do Certificates Come In?

The NJE Bridge and JES2/JES3 NJE implementations have the ability to connect using the SSL protocol. However, both servers need what is called an SSL certificate to be able to establish a secure connection.

### What is an SSL Certificate and How Does it Work?

SSL certificates have a key pair: a public and a private key. These keys work together to establish an encrypted connection. The certificate also contains what is called the "subject", which is the identity of the certificate owner.

To get a certificate, you must create a Certificate Signing Request (CSR) on your server. This process creates a private key and public key on your server. The CSR data file that you send to the SSL certificate issuer (called a Certificate Authority or CA) contains the public key. The CA uses the CSR data file to create a data structure to match your private key without compromising the key itself. The CA never sees the private key.

Once you receive the SSL certificate, you install it on your server. You also install an intermediate certificate that establishes the credibility of your SSL certificate by tying it to your CA's root certificate. The instructions for installing and testing your certificate are

based on the OpenSSL implementation of SSL; instructions for managing OpenSSL are available in many locations online.

In Figure 7, you can see what is called the certificate chain. It connects your server certificate to your CA's (in this case DigiCert's) root certificate through an intermediate certificate.

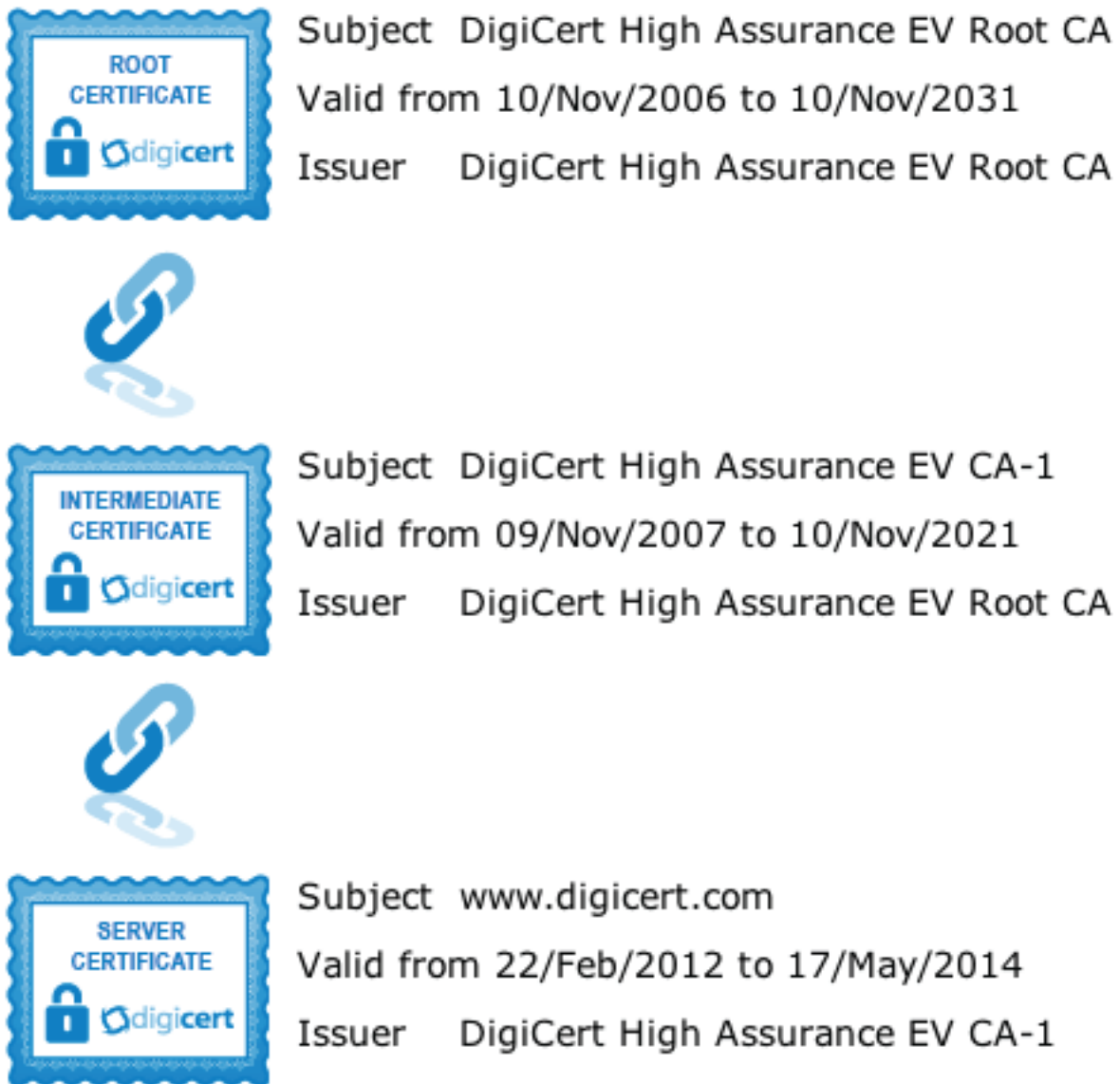


Figure 7. Intermediate Certificate Chain

The most important part of an SSL certificate is that it is digitally signed by a trusted CA. Anyone can create a certificate, but SSL implementations only trust certificates that come from an organization on their list of trusted CAs. OpenSSL comes with a pre-installed list of trusted CAs, known as the Trusted Root CA store. In order to be added to the Trusted Root CA store, and thus become a widely-trusted certificate authority, a



company must comply with and be audited against security and authentication standards established by application vendors.

An SSL certificate issued by a CA to an organization and its domain/website verifies that a trusted third party has authenticated that organization's identity. Since the browser trusts the CA, the browser now trusts that organization's identity too. The browser lets the user know that the website is secure, and the user can feel safe browsing the site and even entering their confidential information.

## How Does the SSL Certificate Create a Secure Connection?

When a client attempts to access a server that is secured by SSL, the client and the server establish an SSL connection using a process called an "SSL handshake". Note that the SSL handshake is invisible to the servers and happens invisibly.

The SSL handshake is implemented in 5 steps:

1. The client connects to a server, and requests the server's identity.
2. The server sends a copy of its SSL certificate, including the server's public key.
3. The client checks the certificate's root against a list of trusted CAs, and that the certificate is unexpired or unrevoked. It also checks that the common name is valid for the site that it's connecting to. If the browser trusts the certificate, it sends back a symmetric session key using the server's public key.
4. The server decrypts the symmetric session key using its private key, and sends back the acknowledgement to start the encrypted session.
5. The two sides (server and client) now encrypt all transmitted data with the session key.

Essentially, three keys are used to set up the SSL connection: the public, private, and session keys. Anything encrypted with the public key can only be decrypted with the private key, and vice versa.

Because encrypting and decrypting with private and public key takes a lot of processing power, they are only used during the SSL handshake to create a symmetric session key. After the secure connection is made, the session key is used to encrypt all transmitted data.

---

## SSL Configuration Files

In the NJE Bridge implementation, the following three files control the process of encrypting the connection. Each SSL-encrypted line should have a unique .crt and .key file pair identifying the link. The njeCA.crt file is used when an external trusted certificate authority is not available or you do not wish to use an external CA.

All of these files should be located in the directory named by the CERTDIR statement in nje.cf. See "CERTDIR (Set Path for SSL-related Information)" on page 26 for information on the certificate directory.

## nodename.crt

This file contains the individual certificate for this nodename. The certificate is supplied by your certificate authority, based on the certificate signing request you sent. Any filename will work, but the extension should remain .crt. The filename (without the .crt extension) should be specified using the CERTFILE keyword in the line definition in nje.cf. See the description of the CERTFILE keyword (“CERTFILE (Set File Name of Certificate ID For This Node)” on page 27) for how to specify the certificate file basename.

Certificates and keys can be in any format supported by OpenSSL.

## nodename.key

This file contains the private key for the certificate stored in nodename.crt. The key is usually issued at the same time as the certificate and is supplied by the certificate authority. To work properly in the NJE Bridge implementation, the filename of the key file and the certificate file must be the same.

Certificates and keys can be in any format supported by OpenSSL.

## njeCA.crt

If you do not have an external CA or do not wish to use one, this file contains a self-signed root certificate that can be used as an internal CA for links within your organization.

**Note**

Self-signed certificates are not recommended for links outside your company or over the public Internet. If you are communicating with outside entities, we strongly recommend you use certificates signed by a commercial CA that is included in OpenSSL's default trust list.

---

## Certificate Tasks

The SSL implementation in the NJE Bridge is based on the OpenSSL implementation. All certificate processing is done using the OpenSSL utilities. Details of the steps, formats, and commands are located at:

[https://wiki.openssl.org/index.php/Command\\_Line\\_Uutilities](https://wiki.openssl.org/index.php/Command_Line_Uutilities)

## Setting Up SSL with Self-Signed Certificates

The following section walks through a configuration of a SSL-protected link between two nodes, FEDORA16 and CTS6NJE. Figure 8 on page 75 shows the configuration file for node FEDORA16.

Key statements are noted in Figure 8 on page 75 by numbers in brackets. These statements are:

```

NAME          FEDORA16
IPADDRESS     fedora16.devlab.example.net
QUEUE        /var/spool/nje
TABLE        /var/run/nje.route
USEREXITS    /etc/nje/file-exit.cf
MSGEXITS     /etc/nje/msg-exit.cf
LLEVEL       1
DEFFORM      STANDARD
DEFAULT-ROUTE SNAVM4
CERTDIR      /etc/pki/nje [1]
CERTFILE     fedora16.pem [2]
PRIVATEKEY   fedora16.key [3]
GROUP        nje
ADMGROUP     njeadmin
LICENSE      XXXXX-XXXXX-XXXXX-XXXXX-XXXXX

LINE 0 CTS6NJE
TYPE         TCPV4SSL [4]
TCPNAME      cts6nje.devlab.example.net
BUFSIZE     4096
TIMEOUT      3
MAX-STREAMS 7

```

Figure 8. Sample nje.cf Configuration for IPv4 SSL link to CTS6NJE

Note ID	Keyword	Description
[1]	CERTDIR	Specifies the directory where NJE keys/certificates are kept
[2]	CERTFILE	Name of the certificate to be used by NJE SSL connections for this node.
[3]	PRIVATEKEY	Name of the private key associated with the certificate
[4]	TYPE	Line type b for SSL this may be TCPV4SSL or TCPV6SSL

The certificate and private keys are generated by a certificate authority. We are generating a self-signed certificate, so we need to generate these files and keys using a local certificate authority process.

## Generating a Self-Signed Certificate

To use a self-signed certificate - in effect, creating your own Certificate Authority (CA) - then there are a number of steps required. In this example, the CA is being set up on one of the NJE nodes; if you plan a large deployment, it is wise to set up a separate system as your local CA, and generate all your self-signed certificates there. The steps below generate first a CA, and then a certificate for a node.

## Generating a Local Certificate Authority (CA)

1. Modify the openSSL Configuration File

Copy the file `/etc/pki/tls/openssl.cnf` to `/etc/pki/nje/nje.conf` and apply the patch shown in Figure 9 on page 77.

2. Create a directory for newly signed certificates

Use the following command to create the new directory:

```
[root@fedora16]# mkdir /etc/pki/nje/newcerts
```

3. Create a private key for the CA

**Warning:** Keep this key safe as it is the key to the creation of all the certificates you will be using! Possession of this key will compromise all future keys from this CA.

Create the private key by using the following commands:

```
[root@fedora16 private]# cd /etc/pki/nje/private
[root@fedora16 private]# openssl genrsa -out njeCA.key 2048
```

Output from the command should appear similar to:

```
Generating RSA private key, 2048 bit long modulus
```

```
.
.
(lines omitted)
.
.
.
e is 65537 (0x10001)
njeCA.key
```

4. Create the root certificate

The root certificate is going to reside in the `/etc/pki/nje` directory unlike node certificates. To generate the root CA certificate, use these commands (command are wrapped to fit this page, but should be typed as a single line):

```
[root@fedora16 private]# cd /etc/pki/nje/
[root@fedora16 nje]# openssl req -new -x509 -extensions v3_ca -key \
    private/njeCA.key -out njeCA.pem -days 3650
```

5. Complete the CA directory setup by securing the private key and setting the initial serial number of the keys to be issued.

To set the serial number of the keys to be issued, type:

```
[root@fedora16 private]# touch /etc/pki/nje/index.txt
[root@fedora16 private]# echo 1000 >> /etc/pki/nje/serial
```

To secure the CA private key, we ensure the private directory is only accessible by root. The following command performs this task.

```
[root@fedora16 private]# chmod 0400 /etc/pki/nje/private/njeCA.key
```

We have now created a local certificate authority for creating self-signed certificates.

```

--- /etc/pki/tls/openssl.cnf          2012-05-15 14:41:42.000000000 -0400
+++ /etc/pki/nje/nje.cnf            2015-05-06 22:29:55.000000000 -0400
@@ -37,24 +37,24 @@
    default_ca = CA_default          # The default ca section

#####
fll CA_default "

-dir = /etc/pki/CA # Where everything is kept
+dir = /etc/pki/nje # Where everything is kept
certs = $dir/certs # Where the issued certs are kept
crl_dir = $dir/crl # Where the issued crl are kept
database = $dir/index.txt # database index file.
#unique_subject = no # Set to 'no' to allow creation of
# several certificates with same subject.
new_certs_dir = $dir/newcerts # default place for new certs.
-certificate = $dir/cacert.pem # The CA certificate
+certificate = $dir/private/njeCA.pem# The CA certificate
serial = $dir/serial # The current serial number
crlnumber = $dir/crlnumber # the current crl number
# must be commented out to leave a V1 CRL
crl = $dir/crl.pem # The current CRL
-private_key = $dir/private/cakey.pem# The private key
+private_key = $dir/private/njeCA.key# The private key
RANDFILE = $dir/private/.rand # private random number file
x509_extensions = usr_cert# The extensions to add to the cert
# Comment out the following two lines for the "traditional"

```

Figure 9. Patch to the openssl Configuration File

**Generate a Certificate Signing Request for A Node:** This certificate signing request will be used to create the certificate for use by the node for setting up SSL connections with other nodes. This step will create both a certificate signing request (CSR) and a private key for use by the node.

To do this, we type:

```

[root@fedora16]# openssl req -newkey rsa:2048 -nodes \
                    -keyout /etc/pki/nje/private/fedora16.key \
                    -out fedora16.csr

```

Output from this command should appear similar to:

Generating a 2048 bit RSA private key

.  
. .  
. .  
. .

writing new private key to '/etc/pki/nje/private/fedora16.key'

-----

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [XX]: <country>

State or Province Name (full name) []: <state>

Locality Name (eg, city) [Default City]: <city>

Organization Name (eg, company) [Default Company Ltd]: <company name>

Organizational Unit Name (eg, section) []: <division>

Common Name (eg, your name or your server's hostname) []: <base hostname>

Email Address []: <user@example.com>

Please enter the following 'extra' attributes

to be sent with your certificate request

A challenge password []:

An optional company name []:

The file specified in the -out option (fedora.csr in this example) contains the certificate signing request (CSR) to be processed in the next step.

**Sign The Certificate:** We then use the CA certificate to process the CSR and create a certificate that will be used by the node for SSL communication. The following command processes and signs the certificate.

```
[root@fedora16]# openssl ca -config /etc/pki/nje/nje.cnf \  
                           -out fedora16.pem b ss_cert fedora16.csr \  
                           -batch
```

The output from the command should be similar to:

```
Using configuration from /etc/pki/nje/nje.cnf
Check that the request matches the signature
Signature ok
```

Certificate Details:

```
Serial Number: 4098 (0x1002)
Validity
  Not Before: May  7 02:39:28 2015 GMT
  Not After : May  6 02:39:28 2016 GMT
Subject:
  countryName           = <country code>
  stateOrProvinceName  = <state>
  organizationName     = <company name>
  organizationalUnitName = <division>
  commonName           = <hostname>
  emailAddress         = <user@example.com>
X509v3 extensions:
  X509v3 Basic Constraints:
    CA:FALSE
  Netscape Comment:
    OpenSSL Generated Certificate
  X509v3 Subject Key Identifier:
    D3:FB:19...
  X509v3 Authority Key Identifier:
    keyid:2C:4D:0E...
```

```
Certificate is to be certified until May  6 02:39:28 2016 GMT (365 days)
Sign the certificate? [y/n]:y
Write out database with 1 new entries
Data Base Updated
```

The file specified on the `-out` option (in this case, `fedora.pem`) contains the certificate to be used to identify this node.

**Keys and Certificates for Other Nodes** If the other node is another NJE Bridge node, perform the step in “Generate a Certificate Signing Request for A Node” on page 77 to create a CSR and private key for the node, and then copy the CSR to the CA and sign it according to the command in “Sign The Certificate” on page 78.: Copy the signed certificate back to the node where it will be used and create the symbolic links according to the process described in “Create Symbolic Links for Hash of Certificates” on page 80.

If the other node is not a NJE Bridge node (e.g., a z/OS node), consult the documentation for that system to determine the process for generating a CSR. Use the generated CSR from the other system as input to the command in the step described in “Sign The Certificate” on page 78.

**Distributing the Intermediate Certificate Chain:** For our self-signed certificates there are a number of files that need to be shared between all nodes that will use SSL.

**Warning:**

Certificates are public files so there is no security exposure. It is the private key of each node that needs to be kept secret.

Copy all node certificates into `/etc/pki/nje/cacerts`, and the CA certificate into `/etc/pki/nje`.

**Create Symbolic Links for Hash of Certificates:** Once the certificates have been distributed, use the `c_rehash` command on each node to create the necessary symbolic links that openssl will search for when retrieving certificates.

In this example, we have created a second node's certificate (cts6nje) and placed it in the cacerts directory.

```
[root@fedora16]# c_rehash /etc/pki/nje
Doing /etc/pki/nje
njeCA.pem => 3cac39db.0
```

```
[root@fedora16]# c_rehash /etc/pki/nje/cacerts/
Doing /etc/pki/nje/cacerts/
cts6nje.pem => fcc695b9.0
fedora16.pem => 72cb3b4f.0
```

At this point, we have generated all the necessary files to enable SSL encryption of the link.

## Verifying Connection Encryption

Unless there are errors, you should not see any SSL messages appearing in the log.

If you set the logging level to 4 the following messages are typical of a successful connection:

```
FEDORA16 ==> SSL 3.3 [length 0x00e2]
FEDORA16 <== SSL 3.1 Handshake [length 0x0035], ServerHello
FEDORA16 TLS server extension "unknown" (id=65281), len=1
FEDORA16 TLS server extension "server ticket" (id=35), len=0
FEDORA16 <== SSL 3.1 Handshake [length 0x0831], Certificate
FEDORA16 <== SSL 3.1 Handshake [length 0x0009], CertificateRequest
FEDORA16 <== SSL 3.1 Handshake [length 0x0004], ServerHelloDone
FEDORA16 ==> SSL 3.1 Handshake [length 0x0830], Certificate
FEDORA16 ==> SSL 3.1 Handshake [length 0x0106], ClientKeyExchange
FEDORA16 ==> SSL 3.1 Handshake [length 0x0106], CertificateVerify
FEDORA16 ==> SSL 3.1 ChangeCipherSpec [length 0x0001]
FEDORA16 ==> SSL 3.1 Handshake [length 0x0010], Finished
FEDORA16 <== SSL 3.1 Handshake [length 0x04ca]???
FEDORA16 <== SSL 3.1 ChangeCipherSpec [length 0x0001]
FEDORA16 <== SSL 3.1 Handshake [length 0x0010], Finished
```

---

## Setting Up SSL Link Encryption with External Certificates

If you need to set up the NJE Bridge to use externally generated certificates, please contact SNA support for assistance.



---

# File and Message Processing Exits

---

## File Processing Exit (file-exit.cf)

NJE transmissions are directed to a particular user at a specified node, however, the NJE Bridge supports the concept of a "pseudouser" which allows fake NJE destination userid to be mapped to specific programs or services without requiring entries in the password file for those usernames. An example of this implementation is the "echo" service specified in `file-exit.cf`.

The NJE Bridge code also allows for mappings between real userids; for instance, it is common to define messages to NJE userid `MAINT` to be redirected to `root`, as `root` is the Unix world's rough equivalent of the z/VM superuser.

These mappings are performed in the NJE Bridge file `/etc/nje/file-exit.cf`. This file specifies how files to or from particular destinations or NJE userids are to be handled; effectively, it routes special cases for file transfers, including printing.

---

## File Pattern Matching Syntax

The `file-exit.cf` file contains a set of patterns that indicate specific actions to be performed if (or when) the NJE attributes of a file match a pattern. Each line in the file represents a pattern and an accompanying action. (The `file-exit.cf` file will look familiar to users of the z/VM Programmable Operator (PROP); it is conceptually very similar in nature and implementation).

Patterns can be specified as text values (which will be matched exactly, e.g., pattern `ECHO` will match only the value `ECHO`), or as wildcards. A single asterisk (\*) will match any token.

---

## Coding File Patterns And The file-exit.cf File

The contents of the file are separated into two parts: general configuration statements, and patterns, separated by the `Exit-Table:` keyword. The general configuration statements are listed in Table 3 on page 83.

Table 3. General Configuration Statements in file-exit.cf	
Keyword	Description
#	A line with a octothorpe (#) in column 1 is a comment. The remainder of the line after column 1 is not parsed.
;	A line with a semicolon (;) in column 1 is a comment. The remainder of the line after column 1 is not parsed.
Spool-Dir:	Indicates the location of the NJE spool directory on this system. The default value is /var/spool/nje. Do not change this value without consulting with SNA support.
Postmast-Dir:	Indicates the location of the directory representing the NJE user POSTMAST (or the userid acting as the site administrator), with the NJE userid in all capital letters. The default value is /var/spool/nje/POSTMAST and should not be changed without consulting SNA support.
Exit-Table:	Indicates the end of general configuration statements. Any following lines are interpreted as patterns according to Table 4 on page 84.

Statements following **Exit-Table:** are considered patterns. Each line in the table is a single (long) line, containing the following fields. Each field provides a set of matchable characteristics, from which the NJE Bridge can determine how to appropriately route the file. The patterns are columnar in nature, and the columns can be interpreted according to the entries in Table 4 on page 84.

<b>Parameter/Keyword/Token</b>	<b>Description</b>
TOUSER	This is the 1-8 character user name of the file recipient. This may or may not correspond to an actual user on your system; file-exit.cf allows you to create pseudo-users such that a user name may, rather than triggering a delivery attempt, trigger some other action.
TONODE	This is the 1-8 character node name of the file recipient. Although intended for routing messages to other nodes, this too can be trapped and trigger arbitrary actions. Note that, like the node name in nje.cf, the default of NJENODE1 here should be globally changed to your own nodename.
FNAME	This is the 1-8 character file name of the received file.
FTYPE	This is the 1-8 character file type of the received file.
PUN?	If this is set to PUN or PUNCH, it matches punch files. If it is set to PRT, PRI, PRIN or PRINT or SYSOUT, it matches print files. If it is set to JOB or SYSIN, it matches incoming files with the SYSIN flag set, and if set to *, all files.
CLASS	This is a classification mechanism for incoming files. For example, PUN files of class M and destined for user MAILER are taken (by convention) to be mail files and will be turned into SMTP mail by the program "mailfy."
FRUSER	This is the 1-8 character userid of the originating user for the file.
FRNODE	This is the 1-8 character node name of the node the file came from.
DIST	This is the 1-8 character version of the NJE distribution code tag. This is an arbitrary tag that the sender sets to allow classification of transmitted files.
FORM	This is the 1-8 character form name.
SPOOLDIR	This is usually left at "default" (and thereby specified by the Spool-Dir: directive); however, it can be modified here to provide relocation to an alternate location based on any of the other fields.

Table 4 (Page 2 of 2). Columns and Pattern Entries in file-exit.cf	
Parameter/Keyword/Token	Description
ACTION	<p>The ACTION field must be one of DISCARD, KEEP, NOTIFY, or RUN.</p> <ul style="list-style-type: none"> <li>• DISCARD takes no arguments, and simply throws away the message.</li> <li>• KEEP takes no arguments; it places the message either into the default spool directory, or into one specified in the action line.</li> <li>• NOTIFY sends a message to an NJE user at the local or at a remote node.</li> <li>• RUN takes two arguments: the name of an arbitrary program to run, and then a string which is passed as the program's argument string.</li> </ul> <p>RUN executes the run program as the recipient user if that is a real user, or as root otherwise.</p> <p><b>CAUTION:</b>  <b>Be careful when specifying arguments to RUN, as these can quite easily run as the superuser. It is especially unwise to uncritically submit JOB files for execution. To preserve system integrity, it is recommended that you create a real user for entries in file-exit.cf that execute applications. This permits you to control the execution context of the application specified on the RUN statement.</b></p>
Extra Arguments	<p>These will specify either the notification recipient, for a NOTIFY action, or the program to run and its arguments for a RUN action.</p>

A fairly extensive example of an application run from file-exit.cf is the interface for printing located in /usr/bin/rprint. rprint is a Bourne shell script that receives a print file and queues it to be printed to a CUPS-managed printer (options are drawn from the forms files). Refer to “Printing” on page 90 for more information on printing and “rprint (Submit Print Jobs to Local Printing System)” on page 97 for more information on rprint.

---

## Message Processing Exit (msg-exit.cf)

Just as `file-exit.cf` handles special-case disposition of incoming or outgoing files, `msg-exit.cf` does the same for messages and commands. Incoming messages or commands can be redirected to specific programs using the `RUN` directive in `msg-exit.cf`.

Incoming messages can be delivered as commands or as interactive messages. The transport mechanism in the NJE protocol is the same, however commands carry a flag to indicate the message is a command and receives special processing. In the `msg-exit.cf` file, commands are identified by a `C` in the `C` column. Interactive messages are identified by a `M` in the `C` column.

---

## Command Pattern Matching Syntax

Token matching in `msg-exit.cf` is processed as follows.

- A single asterisk (\*) surrounded by spaces matches any token, but not a blank.
- A pair of asterisks (\*\*) surrounded by spaces match any number of tokens, including zero tokens.
- An asterisk in the middle of a word specifies a mandatory prefix and then an optional suffix. For example, `TO*KEN` will match any of `TO`, `TOK`, `TOKE`, or `TOKEN`, and no other string.
- Remap pattern `$n` substitutes the `n`th text token into its place
- Any other string is treated as a literal.

---

## Interactive Message Pattern Matching Syntax

The syntax of an interactive message pattern is more similar to the syntax used in `file-exit.cf`. The NJE origin and destination addresses can be matched, and the message can either be discarded or processed by a user-written application. The patterns shown below in Table 7 on page 88 explain the syntax of a message pattern.

---

## Structure Of The msg-exit.cf File

Similar to `file-exit.cf`, the `msg-exit.cf` file is divided into general configuration statements and individual patterns that invoke actions on the incoming message. These statements are documented in detail in the following tables.

## General Message Processing Configuration Statements

The following general configuration statements are located in `msg-exit.cf`.

Parameter/Keyword/Token	Description
CmdHelpFile:	<p>This, by default, points to the file <code>cmd-help.txt</code> which is located in the same directory as the configuration files (default: <code>/etc/nje</code>).</p> <p>Change the location of this file and replace it with your own help text if you want to change it. The contents of this file is sent line-by-line as interactive messages, and should be formatted in a way suitable for display on a terminal. Each line is prefixed by a normal interactive message header, so lines should be limited to 55-60 characters.</p>

A blank line follows the general configuration statements, separating the general configuration statements from the patterns.

## Command Processing Patterns

Each line in the file following the general configuration statements in Table 5 is a pattern. The patterns for command processing are supplied in columns and can be parsed using Table 6.

Token	Description
TOUSER	This represents the 1-8 character destination user name on the incoming command message. For commands, this parameter is not processed, as a command message is always processed by the system receiving the command (and is usually not transmitted by the originating system).
TONODE	This is the 1-8 character destination node name on the incoming message. The default of <code>NJENODE1</code> supplied in the sample <code>msg-exit.cf</code> file installed with the software should be globally changed to reflect your own node name.
FRUSER	This is the 1-8 character user name of the user that originated the message.
FRNODE	This is the 1-8 character node name of the node that originated the message.
C	For commands, this column contains the literal "C".
ACTION	For commands, the ACTION column may be either <code>BUILTIN</code> or <code>RUN</code> .

Table 6 (Page 2 of 2). Command Message Processing Configuration in msg-exit.cf	
Token	Description
additional arguments	<p>Additional argument tokens for BUILTIN are: HELP, HARDCODED, ERROR, and ALIAS.</p> <ul style="list-style-type: none"> <li>• HELP and ERROR may have message strings appended to them. This message string will be displayed to the user.</li> <li>• ALIAS takes a remap-pattern as its argument; this changes one command into another for the consumption of the message exit. See the CPTIME example in the sample msg-exit.cf file.</li> <li>• HARDCODED entries respond with output similar to the VM implementation of NJE (RSCS). These entries are commonly used by link monitoring software, are required, and should not be changed except by consultation with SNA support staff.</li> </ul> <p>Additional argument tokens for RUN are the command line to be executed when the command is received.</p>

## Message Processing Patterns

For interactive messages, the pattern lines can be parsed as in Table 7.

Table 7 (Page 1 of 2). Interactive Message Processing Configuration in msg-exit.cf	
Token	Description
TOUSER	This represents the 1-8 character destination user name on the incoming message.
TONODE	This is the 1-8 character destination node name on the incoming message. The default of NJENODE1 should be globally changed to reflect your own node name.
FRUSER	This is the 1-8 character user name of the user that originated the message.
FRNODE	This is the 1-8 character node name of the node that originated the message.
C	For interactive messages, this column contains the literal "M".



Table 7 (Page 2 of 2). Interactive Message Processing Configuration in msg-exit.cf	
Token	Description
ACTION	<p>For interactive messages, the ACTION may be one of BRCAST, DISCARD, or RUN.</p> <ul style="list-style-type: none"> <li>• BRCAST - The message is sent to all logged in instances of the destination user. This option is useful only for actual human users.</li> <li>• DISCARD - The message is discarded.</li> <li>• RUN - Run the specified command line with the options specified in the additional arguments column.</li> </ul>
Additional Arguments	<p>Additional arguments appended to the RUN command indicating the command line to be run when the message is received.</p> <p>Argument tokens known to RUN are: TOUSER, TONODE, FRUSER, FRNODE, and TEXT:. these can be substituted in the command string by prepending their name with a dollar sign. TEXT is simply the text of the received message. See the ECHO pseudo-user sample entry in the msg-exit.cf file.</p>

No error checking or monitoring is done on RUN commands: from the NJE IP Bridge's perspective, it is purely fire-and-forget, so the msg-exit.cf maintainer should be aware of this fact and handle error conditions appropriately in the executed program.

**CAUTION:**

**Commands will be run as the superuser, and therefore the msg-exit.cf maintainer should be very careful indeed about what exit commands do with the parameters passed to them.**

---

# Forms, Printing and Character Set Translation

---

## Printing

By convention, a print file sent to destination user SYSTEM is assumed to be a file to be printed on the default system printer in portrait mode. Form codes are used to apply additional or special parameters to a print job.

When a print file arrives, it is handed off to the rprint command. See the command reference for rprint (“rprint (Submit Print Jobs to Local Printing System)” on page 97) for details on rprint. The NJE Bridge ships with SYSTEM.STANDARD as a blank file, and SYSTEM.STANDARD functions as a catchall default for unknown forms: if a form is unknown, it will be fed to the local printing system's default print queue with default options.

## Defining A Printer

Printers are defined as destination userids on your NJE Bridge system. This allows all NJE implementations to use common addressing conventions without special handling. Any printer defined to the NJE Bridge node printing system can be used as a NJE-accessible printer.

To add the printer for NJE use, add a line similar to the sample PRTR1 entry in file-exit.cf and change the first token in the line to the NJE name to be used for the printer and the second token to your NJE node name (for example, if you want the printer to be accessible to other NJE users as node SNAFU, printer id YOUBET and be able to refer to it in your JCL as DEST=SNAFU.YOUBET, then the first token in the line should be YOUBET, and the second token should be SNAFU). Change the parameter following the -P to reflect the CUPS printer name for the desired printer.

### Note

If you are directing printed output to an application, this should also be the method you use. You should also encourage users (if their platform supports it) to use DESTIDs defined in their system startup configuration with both NJE node AND userid included - the DESTID can then later be modified if the application moves without requiring JCL changes.

## Defining A Default Printer

Some remote sites may default to using JCL specifying only the node and be unable or unwilling to change it. Defining a printer named SYSTEM will generally capture SYSOUT sent without a complete NJE address. The example file-exit.cf file includes an entry for SYSTEM; you should update this entry to reflect the CUPS printer name you wish to use as the default printer. You may wish to modify the SYSTEM.STANDARD form file to a reasonable default configuration for your site (usually a -o cpi=12 -o landscape entry in form SYSTEM.STANDARD will do something reasonable with random print and punch files).

---

## Forms

Traditionally, a forms code in the IBM implementation indicated which pre-printed paper form the output should be printed on, and the system waited for an operator to mount the correct paper form and tell the printing system to continue. In modern implementations, forms are generally flashed on pages and output is positioned in blanks left in the flashed image. Forms in the NJE Bridge provide a way of specifying a set of particular set of printing options to implement the modern interpretation, which are interpreted by the printing system on the node running the NJE Bridge.

For most operating systems, the printing system is the Common Unix Printing System (CUPS), front-ended by a personality module to emulate either the Berkeley lpr/lpd or System V print semantics. CUPS provides substantially more function than either Berkeley or System V printing; the personality modules are to provide compatibility with existing applications using either of the conventions. Either native printing system (BSD lpr or SVID print) will work with the NJE Bridge (it assumes only that 'lpr' or 'lp' exist in the default system path), but use of the CUPS back end provides a vastly larger set of processing options and printer support, and should be used wherever possible.

### Creating a Form File

With the NJE Bridge, forms are implemented by creating files in `/etc/nje/forms` named `<linkname>.<formname>` (the filename and formname should be in upper-case letters). The file should contain the options needed on your local printing system to implement the print processing you need as you would type them on the command line.

If you are not sure what options your printer is capable of (and you are using CUPS), you can ask CUPS for all the supported options for that printer (we use 'printer2' in this example) by using the following command:

```
lpoptions -p printer2 -l
```

CUPS will respond with all the defined options for that printer and the valid values for those options. Some common option sets are:

```
-o cpi=12 -o landscape -o lpi=6    (gives 132 col landscape output)
-o cpi=17 -o landscape -o lpi=6    (gives 255 col landscape output)
```

If your printer supports duplex printing, adding the `sides=two-sided-long-edge` option will print output on both sides of the paper. Reviewing the entries in the RSCS documentation for form naming conventions used to select printing options may be helpful as well.

CUPS also supports the concept of an option set (an "instance" in the CUPS documentation). If you have instances configured in CUPS, use `-P<printer>/<instance>` as the only option in the forms file. This will likely allow more complex configurations than can be coded normally, especially if the printer is controlled by a desktop system (note that the Internet Printing Protocol is natively supported by desktop and server systems acting as print servers in Windows XP and later, and any Macintosh system (including desktop systems), allowing CUPS to support any printer connected to almost all common environments).

Detailed documentation is installed with CUPS, but is available online at <http://cups.org/documentation.php>.

## SYSTEM.STANDARD (The Default System Form)

The default system form file supplied on install does nothing at all to the file (which usually results in the file being printed in portrait mode, 80 columns, 10 character per inch in Courier). On some platforms, the default has been modified to reflect a useful replacement for line printer output (landscape, 12 cpi, 6 lpi).

In order, the NJE Bridge will:

1. Look for a form file in `/etc/nje/forms` that matches the `userid/linkname` and form code on the file exactly,
2. Look for a `linkname.STANDARD` form file
3. Look for a `SYSTEM.<defform>` form file
4. Look for a `SYSTEM.STANDARD` file as a last-ditch attempt.

As most IBM systems produce 132 column output as `SYSOUT`, it is wise to at least add the `landscape` option to `SYSTEM.STANDARD`. Output that does not fit on a page is truncated and lost.

## Specifying Binary File Processing

In some cases, the originating site has already formatted the output for the printer and the contents of the print file needs only to be passed directly to the printer without translation or other formatting. The form file also provides a method to specify which options are passed to receive when the file is received. Inserting a line in the form file with the "@" character in column 1 of the line indicates that the remainder of the line following the @ character should be passed to the receive command. For example, if the file should be passed without translation (e.g., is binary output ready to be printed), adding the line:

```
@ -b
```

indicates that the file should be passed directly to the printer without translation.

---

## Character Set Translation

Character set translation is applied only at the time of sending or receiving a file at the local NJE Bridge node. Files in transit through the node for other destinations are not modified.

The default translate table is hardcoded in the code and may be replaced or changes by a file supplied by the `EBCDICTBL` statement in `nje.cf` ("EBCDICTBL (Set Path for EBCDIC to ASCII Translation Table File)" on page 31). The table is applied from ASCII to EBCDIC for files sent from the NJE Bridge node, and EBCDIC to ASCII for files received from other NJE nodes. If no specific table is specified, the US default ASCII to EBCDIC table specified by the z/VM TCPIP implementation is used.

## Coding a Custom Translation Table

To supply a custom translation table, you will need to create a file in the following format. A valid translate table has 3 lines:

1. An eyecatcher of the form ASC<=>EBC.
2. A 256 byte line indicating the EBCDIC character to be output for each incoming ASCII character.
3. A 256 byte line indicating the ASCII character to be output for each incoming EBCDIC character.

In the 256 character lines, you must supply a character for each position to be translated. An editor capable of accepting input in hex characters may be helpful in creating the translation mappings.

## Specifying Your Custom Table

Use the `EBCDICTBL` statement to provide the full Unix path to the desired translate table. Note that this table is system-wide; if you have a need to provide user-specific translation tables, please contact SNA support to open a development request.

## Specifying Binary Data (No Translation)

Most of the user commands have a `-b` option to specify that the file is binary data and should not be translated. See the individual command entries in the End User Command Guide for end user command options and syntax.



---

# Administrative Commands and Tasks

---

## Administrative Commands

Several administrator-only commands are included to help manage the NJE Bridge environment. Each command is listed on a following page. Users running these commands must belong to the NJE administrative group or the command will exit immediately.



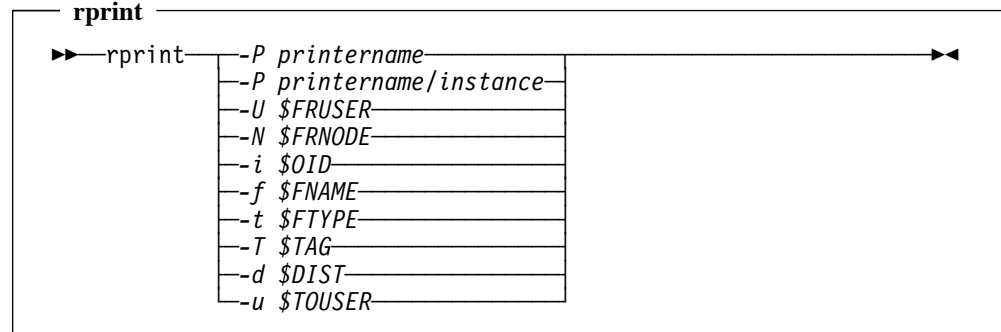
---

# rprint (Submit Print Jobs to Local Printing System)

## Purpose

The rprint utility accepts printed output from NJE originating userids and passes it to a designated Unix printer, converting and respooling the print job as necessary. Processing options for print jobs are specified by the form code supplied by the originating userid.

## Format



## Parameters

Note that all parameters to rprint are case-sensitive.

### **-P printername**

Specify the Unix printer id to spool NJE output to. This option is case-sensitive and should match the name defined in CUPS or an alternative printing system.

### **-P printername/instance**

If using CUPS, specify the name of a printer and a predefined option set in CUPS created using lptions. Refer to the CUPS documentation for information on creating instances.

### **-U \$FRUSER**

Supply the value of the originating userid from the NJE header of the file.

### **-N \$FRNODE**

Supply the value of the originating NJE node name from the NJE header of the file.

### **-i \$OID**

Supply the original NJE spool file id number to rprint.

### **-f \$FNAME**

Set the file name of the file to be printed (in CMS format).

### **-t \$FTYPE**

Set the file type of the file to be printed (in CMS format).

### **-T \$TAG**

Supply the NJE file tag string to rprint.

### **-d \$DIST**

Supply the NJE distribution code to rprint.

### **-u \$TOUSER**

Supply the destination userid of the file (used as the NJE printer identifier).

## Usage

See the supplied file-exit.cf in /etc/nje for usage examples.

## Examples

See the supplied file-exit.cf in /etc/nje for usage examples.

## Comments

1. Options and variable names shown above are case-sensitive and should be written as shown.
2. rprint is only an example implementation. You can replace it with any processing you wish.

**Note**

If you are implementing archival services or other print processing, you may wish to use rprint as a shell to implement your own interface.

---

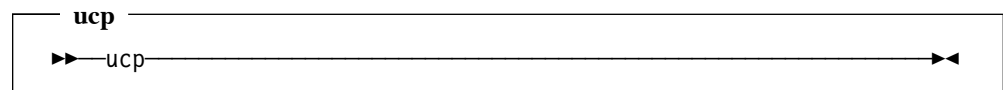
# ucp (User Control Panel)

## Purpose

ucp provides a method for controlling the operation of the NJE Bridge daemons while the application is running. ucp allows users to start and stop lines, query the state of the application, modify routing tables and debugging levels, and initiate a graceful shutdown of the application.

Users intending to use the ucp command must be the superuser or a member of the Unix group listed in the ADMGROUP statement (default "njeadmin") in nje.cf.

## Format



## Parameters

ucp has no command line arguments. All operations are performed by issuing subcommands from an interactive prompt inside the ucp command environment. The ucp subcommands are documented individually in “UCP Subcommands” on page 100.

## Usage

Use the ucp command whenever you wish to modify the operation of the NJE daemons without restarting them. See the pages for individual commands in the following section “UCP Subcommands” on page 100 for details of the features available.

### CAUTION:

**This command can radically affect NJE operation. Restrict access to the ucp command to avoid unexpected outages or problems.**

## Examples

An example of running ucp against a live system might appear similar to the following (example lines are wrapped for clarity):

```
root@vsrv053 # ucp
NJE-CP> show line
NJE-CP>
NJE0098I Line 0 PRODHUB Type: TCPv4 Status: RETRYING
          Files In/Out: 0 0 Bytes In/Out: 264 354 Buffer Size: 4096
NJE0098I Line 1 DEVHUB Type: TCPv4 Status: ACTIVE
          Files In/Out: 0 0 Bytes In/Out: 672 1022 Buffer Size: 4096
```

## Comments

1. See “ADMGROUP (Set Unix Group Name for Administrative NJE Users)” on page 23 for details about group membership requirements for administrative commands.

---

## UCP Subcommands

ucp implements a number of subcommands. The following pages document each subcommand and function.

---

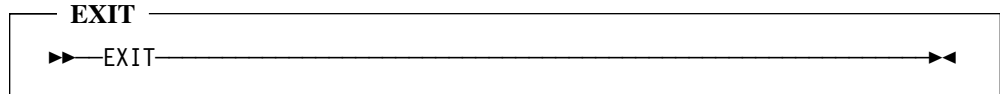
# EXIT

## (Terminate Interactive UCP Session)

### Purpose

The EXIT subcommand ends the ucp session.

### Format



### Parameters

This subcommand has no parameters.

### Usage

The subcommand ends a interactive ucp session and returns you to the command line prompt.

### Examples

Exiting ucp appears like this:

```
NJE-CP> exit
root@wizard#
```

### Comments

There are no comments on this subcommand.

---

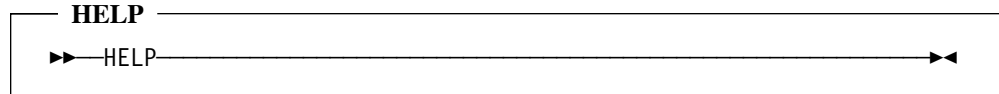
# HELP

## (Display UCP Help File)

### Purpose

The HELP subcommand displays abbreviated help information.

### Format



### Parameters

There are no parameters for this subcommand.

### Usage

This subcommand displays an abbreviated help file. Refer to this document for detailed explanation of the ucp subcommands.

### Examples

The abbreviated help information displayed by the HELP subcommand is similar to:

```
NJE-CP> help
HELP - Show this message
SHOW LINE/QUEUE - Show lines or queue status
SHOW ROUTE [<node>] - Show routing information for all or <node>
START LINE n - Start a closed line
SHUT [ABORT] - Shutdown or abort the whole program
STOP LINE - Stop a line
RESCAN EXITS - Rescan file and message exits.
RESCAN QUEUE - Rescan queue and requeue files.
RESCAN ROUTE - Reopen route database.
LOGLEVEL n - Set the loglevel to N
ROUTE xxx TO yyy - Change the routing table.
Route to OFF will delete the route entry.
EXIT/QUIT - Exit this utility.
NJE-CP>
```

### Comments

There are no comments for this subcommand.

# LOGLEVEL

## (Change Logging Detail Level)

### Purpose

The LOGLEVEL subcommand sets the detail level of logging performed during NJE operation.

### Format

```
LOGLEVEL level
```

### Parameters

#### level

Detail level of logging. Valid values are:

Table 8 (Page 1 of 2). Logging Detail Levels	
Log Detail Level	Description
0	No diagnostics. Effectively equivalent to log detail level 1.
1	Normal operation. A message is logged for each successfully completed file transmission or receipt on all operational lines. Errors or line state transitions (up/down, error) are logged.
2	Job progress logging. Messages are logged for each NJE dataset header and trailer processed.
3	Extended job progress logging. Messages are logged for line and stream selection processing in addition to the job progress logging noted for detail level 2.
4	Display all NJE internal state machine processing.  <b>CAUTION:</b> <b>This detail level can produce voluminous output and may rapidly fill log disks and/or overwhelm central logging servers. Do not run for extended periods with this level of detail enabled.</b>
5	Reserved for future use.

Table 8 (Page 2 of 2). Logging Detail Levels	
Log Detail Level	Description
6	<p>Trace all data streams and display hex dump of traffic stream.</p> <p><b>CAUTION:</b>  <b>This detail level produces voluminous output and will dramatically impact performance of the daemons. The quantity of output produced will rapidly fill log disks and/or overwhelm central logging servers. Do not run for extended periods with this level of detail enabled unless requested by SNA support.</b></p>

## Usage

This is the preferred method of varying the log level. The LLEVEL keyword in nje.cf can also vary the default detail level of logging, however use of the ucp LOGLEVEL subcommand to dynamically vary the log detail level during operation allows the default level at daemon startup to always revert to a safe level of detail while permitting intensive logging to be enabled as required.

## Examples

This subcommand produces no visible feedback at the command line.

## Comments

1. Use this command with caution (see above). Operating with elevated log levels impacts performance for all lines and all file processing.
2. In general, the output of log detail levels greater than 1 are intended for SNA support, and will be unintelligible to end users. Output from extended logging should not be considered a programming interface and may be changed without notice.



---

# QUIT

## (Terminate Interactive UCP Session)

### Purpose

This subcommand is a synonym for EXIT. See “EXIT (Terminate Interactive UCP Session)” on page 101 for the description of the EXIT subcommand.

### Format

<pre>QUIT</pre>
-----------------

### Parameters

This subcommand has no parameters.

### Usage

The subcommand ends a interactive ucp session and returns you to the command line prompt.

### Examples

Exiting ucp appears like this:

```
NJE-CP> quit  
root@wizard#
```

### Comments

There are no comments on this subcommand.

---

# RESCAN EXITS

## (Rescan File and Message Exit Files)

### Purpose

The RESCAN EXITS subcommand re-reads the file and message processing exit files if they have been updated while the daemon is running.

### Format

<b>RESCAN EXITS</b>
▶—RESCAN EXITS—▶

### Parameters

There are no parameters for this subcommand.

### Usage

RESCAN EXITS allows you to update the exit definitions files without interrupting other processing. It is frequently used when testing new file or message processing scripts. See “File Processing Exit (file-exit.cf)” on page 82 for more information on file and message processing.

### Examples

This command has no visible output.

### Comments

There are no comments on this subcommand.

---

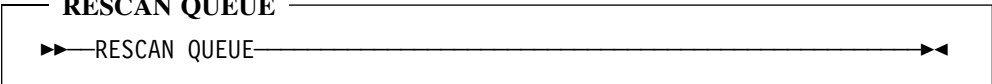
## RESCAN QUEUE (Rescan All File Queues and Reschedule Files)

### Purpose

The RESCAN QUEUE subcommand rescans the queue according to the current routing tables and system settings and requeues files as appropriate.

### Format

```
RESCAN QUEUE
```



### Parameters

There are no parameters for this subcommand.

### Usage

RESCAN QUEUE forces the daemon to re-evaluate file transmission and line scheduling for the current queue of files to be sent. In a large network this process can be quite time-consuming, and this command allows initiating the process without disrupting other processing.

This subcommand is also helpful when NJE routing has been changed (to avoid a failed system, for example) and files need to be rescheduled on other lines to bypass the failed system.

### Examples

This subcommand has no visible output.

### Comments

There are no comments on this subcommand.



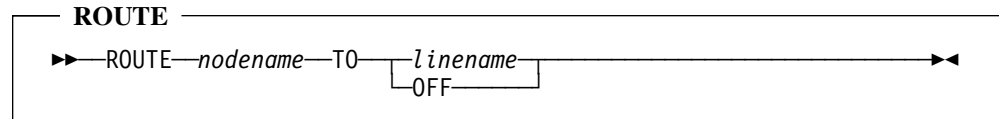
---

## ROUTE (Modify NJE Routing Database)

### Purpose

The ROUTE subcommand permits changing the NJE routing database during operation.

### Format



### Parameters

#### **nodename**

A valid NJE node name for which the routing is being defined.

#### **linkname**

The name assigned on the LINE statement describing the connection to a remote host. The LINE statement for the host must exist in `nje.cf` before it can be used with the ROUTE subcommand. Refer to “LINE (Define A Connection to Another NJE Host)” on page 51 for more information on the LINE statement.

#### **OFF**

Delete the routing information for the specified nodename.

### Usage

Changes made with the ROUTE subcommand are made to the in-memory and on-disk copies of the routing tables, and take effect immediately at the next opportunity for file or message routing. Changes made to the on-disk copy are persistent across system restarts.

In small networks, the ROUTE subcommand is the preferred method for updating routing tables. If your network is large (greater than 50-60 nodes), you may wish to consider generating routing databases centrally, updating the routing database files on disk, and using RESCAN ROUTES to commit the updated routing file into production use.

### Examples

To indicate that NODE3 should be routed via line LINK2, the ROUTE statement might appear similar to:

```
NJE-CP> ROUTE NODE3 TO LINK2
NJE-CP>
```

All traffic for NODE3 will now use LINK2.

To redirect traffic for NODE3 to LINK4, you can update the route by issuing:

```
NJE-CP> ROUTE NODE3 TO LINK4
```

Future traffic will now use LINK4 to communicate with NODE3.

To remove the routing for HOPELESS from the routing database, issue:

```
NJE-CP> route hopeless to off
```

Node HOPELESS will be removed from the routing tables and become effectively unreachable from this node (unless HOPELESS can be reached by the node specified by a DEFAULT-ROUTE statement in `nje.cf`).

## Comments

1. All NJE node names in use in a network must be defined at all nodes in the network. The DEFAULT-ROUTE keyword (see “DEFAULT-ROUTE (Set Default NJE Route if No Specific Route Supplied)” on page 28) mitigates this requirement for leaf nodes with single links to a more well-connected NJE host.
2. In networks with PATHMGR YES NJE Bridge nodes or JES2 nodes with PATHMGR=Y nodes, use of line resistance values can affect line selection in a way that overrides explicit routing. Contact SNA support if you suspect files are being routed sub-optimally.

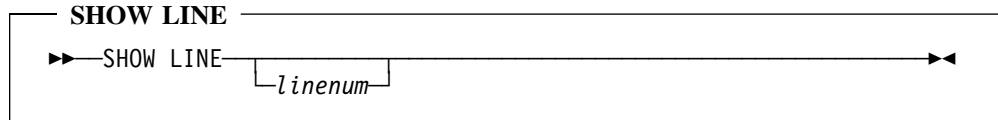
# SHOW LINE

## (Display Defined Lines and Line Status)

### Purpose

The SHOW LINE subcommand displays the defined lines and their current status. If a specific line number is provided, the state of the indicated line is displayed.

### Format



### Parameters

#### linenum

Unique line number for the line to be displayed. This parameter is optional; if omitted, all lines defined for this NJE node are displayed along with statistics for line activity. The parameters displayed in the resulting message are:

Table 9 (Page 1 of 2). SHOW LINE Parameters	
Parameter/Keyword	Description
NJE0098I	Message id for response.
Line	Indicates beginning of information for a new line entry.
<linenum>	Unique line number for this line from the LINE statement in nje.cf. Refer to "LINE (Define A Connection to Another NJE Host)" on page 51 for more information on the LINE statement.
Type	Keyword delimiting line type information. See "TYPE (Set Type of Connection Method for This Line)" on page 69 for valid line types.
<nodename>	NJE node name of the system on the remote end of the NJE connection from the LINE statement for this connection. See "LINE (Define A Connection to Another NJE Host)" on page 51 for more information on the LINE statement.
Status	Keyword delimiting line status.
<linestatus>	Line status. Valid values are shown in Table 10 on page 112.
Files In/Out	Keyword delimiting file in/out count.

Parameter/Keyword	Description
n n	Number of files in/out, respectively. These counters are incremented when a file is completely received and/or transmitted to a remote NJE node and an acknowledgement of the final record of the file is received/transmitted. If a file is interrupted for some reason, it is not counted here.
Bytes In/Out	Keyword delimiting byte in/out count.
n n	Bytes received/transmitted, respectively. These counters are incremented as records are received and transmitted, and should change frequently. All bytes transmitted and received (including partial transmissions and header information) are counted here.
Buffer Size	Keyword delimiting buffer size information.
<bufsize>	Maximum transmission/receive buffer size for this connection.  Note that this value must match the value specified on the BUFSIZE keyword for this line. Refer to “BUFSIZE (Set Maximum Transfer Buffer Size for This Line)” on page 47 for information on the BUFSIZE line keyword.

Valid values for line status are:

Line Status	Description
INACTIVE	Line is not active.
SIGNOFF	Line shutdown is in progress.
DRAIN	Line is administratively stopped.
ISIGNON	Initial signon in progress. The TCP OPEN/ACK has occurred, and the NJE Bridge is about to send the link signon record.
ACTIVE	Line is in service and processing files.
FSIGNON	The NJE link signon record has been sent, and the link is awaiting a signon response.
LISTEN	Line is awaiting a TCP connection from the remote node.
RETRYING	Line has received an error and is retrying the operation.



Table 10 (Page 2 of 2). Line Status Values	
Line Status	Description
DISABLED	Part of the SSL encryption initialization process has failed. The line is stopped.

## Usage

This subcommand is used to display the state of lines and some frequently useful information about a line. The examples below illustrate some common responses.

## Examples

To display all defined nodes:

```
root@wizard# ucp
NJE-CP> show line
NJE-CP>
NJE0098I Line 0 PRODHUB Type: TCPv4 Status: RETRYING
          Files In/Out: 0 0 Bytes In/Out: 264 354 Buffer Size: 4096
NJE0098I Line 1 DEVHUB Type: TCPv4 Status: ACTIVE
          Files In/Out: 0 0 Bytes In/Out: 672 1022 Buffer Size: 4096

NJE-CP> quit
root@wizard#
```

Displaying a single line might appear similar to:

```
root@wizard# ucp
NJE-CP> show line 1
NJE-CP>
NJE0098I Line 1 DEVHUB Type: TCPv4 Status: ACTIVE
          Files In/Out: 0 0 Bytes In/Out: 672 1022 Buffer Size: 4096

NJE-CP> quit
root@wizard#
```

## Comments

1. In the current release of the NJE Bridge, responses to this command are delivered via NJE interactive messages, which cannot be captured by commands. A future release of the NJE Bridge will provide a method to do this task for scripting purposes.

---

# SHOW QUEUE

## (Show Status of File Transmit/Receive Processing)

### Purpose

The SHOW QUEUE subcommand shows the status of the file transmission and receiving queues on this node. SHOW QUEUE indicates the number of files queued, which line each file is queued on, and other status information about the files.

### Format

```
SHOW QUEUE
```

### Parameters

The SHOW QUEUE subcommand has no parameters.

### Usage

Normal usage of this command displays file queues. If files are accumulating on a specific line, this is usually a signal that processing is impaired on the line, and diagnostics are needed. The output of SHOW QUEUE can be interpreted as follows:

Table 11 (Page 1 of 2). SHOW QUEUE Response	
Token/Keyword	Description
NJE0170I	Message id for response.
<nje-node-name>	NJE node name for the line processing this file. The entry shows the remote node name from the LINE statement.
(<status>)	State of processing for the line. Table 10 on page 112 shows the possible line states.
n	Sequence number assigned to the file in queue. n can be a positive integer from 1 to 9999.
(<filename> <filetype>)	NJE filename and filetype, abbreviated to CMS naming conventions. Filenames from other operating systems will be transformed to meet minimum CMS file naming conventions for display, but the original file naming conventions are preserved within the NJE file transmission headers and are interpreted at the endpoint of the transmission.
Disp	Keyword delimiting the internal dispatch flags for the file. Contact SNA support if you need to interpret this flag.
<dispatch-code>	Table 12 on page 115 contains the valid values for this field. Contact SNA support if you need to interpret this flag.

Table 11 (Page 2 of 2). SHOW QUEUE Response	
Token/Keyword	Description
State	Keyword delimiting the processing state of the file.
<processing state>	Processing state for the file. Table 13 on page 115 contains the valid values for this field.
Size	Keyword delimiting the size of the file.
<size>	Size of file in records.
Org	Keyword delimiting the NJE address of the file sender.
<fromnode.fromuser>	NJE node and userid of file sender.
Dest	Keyword delimiting the NJE destination for the file.
<destnode.destuser>	NJE destination node and userid for the file.

Valid dispatch codes for files are:

Table 12. File Dispatch Code Values	
Dispatch Code	Description
D	Delete file after processing.
H	File is to be held after processing.
K	Keep file after processing.
N	Notify after file received.
R	Run command against file.
W	File is waiting to be processed.
S	File is being transmitted.
X	Discard the file.

Valid file processing states are:

Table 13. File Processing States	
Processing State	Description
WAIT	File is waiting for the associated line to become active.
HOLD	File is on hold pending other processing.
SEND	File is being transmitted.

## Examples

Example output from the SHOW QUEUE command might appear similar to:

```
NJE-CP> show queue
NJE0170I PRINT (INACTIVE)    1 (DCB ASSEMBLE) Disp: D State: WAIT
                               Size: 00000003 Org: SNAVM4.NEALE Dest: PRINT.SYSTEM
NJE0170I PRINT (INACTIVE)    2 (PROFILE EXEC) Disp: D State: WAIT
                               Size: 00000020 Org: SNAVM4.DBOYES Dest: PRINT.SYSTEM
NJE-CP>
```

## Comments

1. In the current release of the NJE Bridge, responses to this command are delivered via NJE interactive messages, which cannot be captured by commands. A future release of the NJE Bridge will provide a method to do this task for scripting purposes.

---

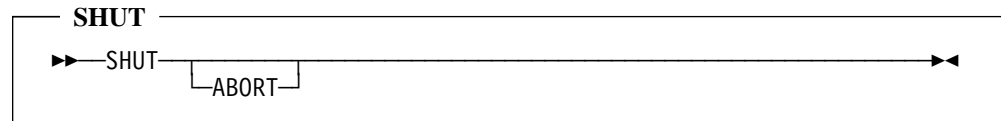
# SHUT

## (Terminate All Connections Gracefully and Halt NJE Processing)

### Purpose

The SHUT subcommand shuts down the NJE IP Bridge.

### Format



### Parameters

#### ABORT

Terminate all connections and end the NJE Bridge immediately. No effort to gracefully shut down connections is made.

### Usage

The SHUT subcommand is used to administratively shut down the NJE daemons during debugging. Normally, the init scripts or `systemctl` should be used to terminate the daemons during system shutdown processing.

### Examples

Aborting the daemon might appear similar to this:

```
NJE-CP> shut abort  
NJE-CP> show line  
NJE0083S Error sending command to NJE subsystem - Connection refused  
NJE-CP>
```

### Comments

1. In the current release of the NJE Bridge, responses to this command are delivered via NJE interactive messages, which cannot be captured by commands. A future release of the NJE Bridge will provide a method to do this task for scripting purposes.

---

## START LINE (Start Inactive Line)

### Purpose

The START LINE subcommand attempts to initiate communication on the designated line. If the line is defined with CONNECT NO, the line begins listening for inbound connections, otherwise this node initiates an outbound connection to the TCP host defined in nje.cf.

### Format

```
START LINE _____  
▶—START LINE—linenum—————▶
```

### Parameters

#### **linenum**

Unique line number for the line to be started. This number is defined by the LINE definition in nje.cf. See “LINE (Define A Connection to Another NJE Host)” on page 51 for more information on line definition.

### Usage

Operators normally use this subcommand to restart processing on a line after an error.

### Examples

Line 1 (SNAVM4) has been stopped due to a processing error. The error has been corrected and we wish to restart the connection. Initially, the stopped connection is displayed as follows (output from SHOW LINE):

```
NJE-CP> show line  
NJE-CP>  
NJE0098I Line 0 DEVHUB Type: TCPv4 Status: ACTIVE  
Files In/Out: 0 0 Bytes In/Out: 223 226 Buffer Size: 4096  
NJE0098I Line 1 SNAVM4 Type: TCPv4 Status: DRAIN  
Files In/Out: 2 0 Bytes In/Out: 2432 73388 Buffer Size: 3976  
NJE0098I Line 2 PRINT Type: TCPv4 Status: DRAIN  
Files In/Out: 0 2 Bytes In/Out: 703 2542 Buffer Size: 4096
```

To restart the connection to SNAVM4, we issue START LINE 1. The resulting output from SHOW LINE appears similar to:

```
NJE-CP> start line 1  
NJE-CP> show line  
NJE-CP>  
NJE0098I Line 0 DEVHUB Type: TCPv4 Status: ACTIVE  
Files In/Out: 0 0 Bytes In/Out: 223 226 Buffer Size: 4096  
NJE0098I Line 1 SNAVM4 Type: TCPv4 Status: ACTIVE  
Files In/Out: 2 0 Bytes In/Out: 2679 73553 Buffer Size: 3976  
NJE0098I Line 2 PRINT Type: TCPv4 Status: DRAIN  
Files In/Out: 0 2 Bytes In/Out: 703 2542 Buffer Size: 4096
```

The connection to SNAVM4 is restarted and functioning.

## Comments

1. In the current release of the NJE Bridge, responses to this command are delivered via NJE interactive messages, which cannot be captured by commands. A future release of the NJE Bridge will provide a method to do this task for scripting purposes.

---

# STOP LINE

## (Gracefully Stop Active Line)

### Purpose

The STOP LINE subcommand performs a graceful halt of processing on the specified line, leaving the line in a administratively halted state. File processing in progress on the line is allowed to complete for any files in transit at the time the STOP LINE subcommand is issued.

### Format

```
STOP LINE linenum
```

### Parameters

#### **linenum**

Unique line number for the line to be stopped. This number is defined by the LINE definition in nje.cf. See “LINE (Define A Connection to Another NJE Host)” on page 51 for more information on line definition.

### Usage

This command is normally used for scheduled outages or debugging purposes when examining connections to a specific host. See the example below for a common usage scenario.

### Examples

Host SNAVM4 is entering a scheduled maintenance window, and file transmission to that host will need to stop while maintenance is underway. To stop file traffic to and from SNAVM4 without interrupting processing for other lines, the following sequence might occur.

Normal operation of SNAVM4 appears similar to the following:

```
NJE-CP> show line
NJE-CP>
NJE0098I Line 0 DEVHUB Type: TCPv4 Status: ACTIVE
          Files In/Out: 0 0 Bytes In/Out: 223 226 Buffer Size: 4096
NJE0098I Line 1 SNAVM4 Type: TCPv4 Status: ACTIVE
          Files In/Out: 2 0 Bytes In/Out: 2679 73553 Buffer Size: 3976
NJE0098I Line 2 PRINT Type: TCPv4 Status: DRAIN
          Files In/Out: 0 2 Bytes In/Out: 703 2542 Buffer Size: 4096
```

To halt processing on line SNAVM4, we issue STOP LINE 1. SHOW LINE then shows that processing has stopped and the line is in a drained state (administratively down). The SHOW LINE output appears similar to:



```
NJE-CP> stop line 1
NJE-CP> show line
NJE-CP>
NJE0098I Line 0 DEVHUB Type: TCPv4 Status: ACTIVE
          Files In/Out: 0 0 Bytes In/Out: 223 226 Buffer Size: 4096
NJE0098I Line 1 SNAVM4 Type: TCPv4 Status: DRAIN
          Files In/Out: 2 0 Bytes In/Out: 2679 73553 Buffer Size: 3976
NJE0098I Line 2 PRINT Type: TCPv4 Status: DRAIN
          Files In/Out: 0 2 Bytes In/Out: 703 2542 Buffer Size: 4096
```

## Comments

1. In the current release of the NJE Bridge, responses to this command are delivered via NJE interactive messages, which cannot be captured by commands. A future release of the NJE Bridge will provide a method to do this task for scripting purposes.

---

# Administrative Tasks

---

## Start A Line/Link

To start a line:

1. Log into the NJE Bridge system as a user with membership in the `njeadmin` group (the superuser is a member of this group).
2. Run `ucp`.
3. Use the `START LINE` command to start the line.
4. Use the `EXIT` or `QUIT ucp` subcommands to exit

---

## Stop A Line/Link

To stop a line:

1. Log into the NJE Bridge system as a user with membership in the `njeadmin` group (the superuser is a member of this group).
2. Run `ucp`.
3. Use the `STOP LINE` command to stop the line.
4. Use the `EXIT` or `QUIT ucp` subcommands to exit

---

## Stop A Line/Link Immediately

To force a line to stop immediately without cleanup:

1. Log into the NJE Bridge system as a user with membership in the `njeadmin` group (the superuser is a member of this group).
2. Run `ucp`.
3. Use the `FORCE LINE` command to stop the line.
4. Use the `EXIT` or `QUIT ucp` subcommands to exit

---

## Display File Queues

To show files being transmitted and received:

1. Log into the NJE Bridge system as a user with membership in the `njeadmin` group (the superuser is a member of this group).
2. Run `ucp`.
3. Use the `SHOW QUEUE` command to show files queued for transmission.
4. Use the `EXIT` or `QUIT ucp` subcommands to exit.

---

## If You Have Trouble

---

# If You Have Trouble

---

## Gathering Information About The Problem

### Logging and Log Files

The application logs information to the general Unix system log using syslog's local0 facility.

### Setting An Elevated Debugging Level

From the ucp control panel tool, use the command LOGLEVEL 6 (see “LOGLEVEL (Change Logging Detail Level)” on page 103 for details of the LOGLEVEL subcommand). This is the highest setting and will log the most debug data.

### What to Capture

To assist in debugging, capture the following files and submit them to support as email attachments when requested.

- The relevant portion of the log file from syslog.

### Turning Debugging Off

From ucp, use the command LOGLEVEL 1 to turn debugging messages back down to normal operational levels. See “LOGLEVEL (Change Logging Detail Level)” on page 103 for information on using the ucp LOGLEVEL subcommand.

---

## Reporting A Problem

Use the instructions supplied to open a trouble ticket per the instructions in your support contract. If requested by the support team, please send the log file demonstrating the problem at that point.

---

## Helpful Hints While Working on the Problem

The following hints will help speed resolution of your issue.

- When reporting a problem, send problem reports to [support@sinomine.net](mailto:support@sinomine.net). You will receive an automated response containing your case number.
- While working with support, please preserve the subject line of the email and ensure that “[sinomine.net #nnnnn]” remains in the first 16 characters of the subject line of your replies. This allows the ticketing system to associate your replies with the problem report automatically, and keeps the flow of conversation connected.
- Please report only one issue per case. Reusing incident numbers confuses issues, and will delay resolution.



---

## Appendix A. Other Helpful References

The NJE protocol is well-documented. The following sections describe where to find more information on the protocol and on the IBM implementations of NJE.

In the following sections, URL references, even if split across multiple lines, are to be entered as a single long string into the browser's location field.

---

### NJE Protocol References

The best reference to NJE as a whole is unquestionably: *z/OS NJE Formats And Protocols* (IBM Document Number SA22-7539)

An earlier version (lacking TCPNJE), IBM Document Number SC23-0070-03, is publicly available at:

- [http://publibz.boulder.ibm.com/cgi-bin/bookmgr\\_OS390/BOOKS/IEA1M503/CCONTENTS?SHELF=IEA1MB00&DN=SC23-0070-03&DT=19980414152716](http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/IEA1M503/CCONTENTS?SHELF=IEA1MB00&DN=SC23-0070-03&DT=19980414152716)

For the initial definition of NJE transmission over TCP/IP, consult Peter Olenick's (of Princeton University) document "Definition of the BITNET II Protocol: A Technical Overview of VMNET". It is available from:

- <http://www.funet.fi/pub/netinfo/BITNET/brf0002.text>

---

### Connecting to IBM Implementations

#### RSCS Connections

For RSCS (including TCPNJE) configuration, use *VM RSCS Planning And Configuration*, IBM Document Number SH24-5219, available at:

- [http://publibz.boulder.ibm.com/cgi-bin/bookmgr\\_OS390/download/DMTA2A03.pdf?DT=19990511213526](http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/download/DMTA2A03.pdf?DT=19990511213526)

#### VSE/POWER Connections

For VSE, there are two primary reference books on NJE configuration. These books are:

- *VSE/POWER Networking*, IBM Document Number SC33-6735  
<http://publibfp.boulder.ibm.com/epubs/pdf/iesnpe20.pdf>
- *VSE/POWER Administration and Configuration*, IBM Document Number SC33-6733  
<http://publibfp.boulder.ibm.com/epubs/pdf/iespme30.pdf>

#### z/OS and JES2 Connections

NJE configuration documentation for JES version 2 (5740-XC6) is available in the following manuals:

- *MVS/SP JES2 Initialization and Tuning*, IBM Document Number SC23- 0065
- *MVS/SP JES2 Commands*, IBM Document Number SC23-0064

- MVS/SP JES2 Logic, IBM Document Number LY24-6008

## **z/OS and JES3 Connections**

NJE documentation for JES3 version 4 (5695-048) is available in the following manuals:

- JES3 V4 Initialization and Tuning Guide, IBM Document Number SC23- 0088
- JES3 V4 Initialization and Tuning Reference, IBM Document Number SC23-0089
- JES3 V4 Commands, IBM Document Number GC23-0090
- JES3 V4 Customization, IBM Document Number LY28-1026
- Diagnosis Reference, IBM Document Number LY28-1032

## **iSeries Connections**

NJE setup examples are best documented in a IBM redbook shown below. The manuals for your release of the iSeries operating system are likely quite dated.

To read more about iSeries NJE implementation, consult:

- VSE/POWER and OS/400 NJE Configuration Guide, IBM document number GG24-4259-00

Note that this document includes only information about the IBM Systems Network Architecture (SNA) implementation of NJE; the utilities and system configurations used to transmit and receive information are the same when used with the NJE Bridge.

---

## Appendix B. Your Comments are Welcome!

NJE Bridge: Configuration and Planning Reference

Use this form to tell us what you think about this manual, or send email to support AT [sinenomine.net](mailto:sinenomine.net). If you have found errors in it, or if you want to express your opinion about how the information is organized or presented, or make suggestions for improvements, this is the form to use.

Be sure to include your name and email address (and optionally, a telephone number with country and area code) if you would like a reply.