# Switch Network Interface for eServer pSeries High Performance Switch Guide and Reference

SC23-4869-04

# Switch Network Interface for eServer pSeries High Performance Switch Guide and Reference

SC23-4869-04

**Note**

Before using this information and the product it supports, read the information in “Notices,” on page 37.

**Fifth Edition (July 2006)**

This edition applies to Switch Network Interface for eServer pSeries High Performance Switch.

A reader’s comment form is provided at the back of this publication. If the form has been removed, address comments to Information Development, Department 04XA-905-6B013, 11501 Burnet Road, Austin, Texas 78758-3400. To send comments electronically, use this commercial Internet address: pserinfo@us.ibm.com. Any information that you supply may be used without incurring any obligation to you.

# Contents

**About This Book** . . . . . v
Who Should Use This Book . . . . . v
Highlighting . . . . . v
Case-Sensitivity in AIX . . . . . v
ISO 9000 . . . . . v
Related Information . . . . . v

**Chapter 1. Overview of Switch Network Interface** . . . . . 1
Switch Network Interface and Multi-Link Support Overview . . . . . 1
User Space Job Support . . . . . 2
Technical Large Pages Overview . . . . . 2
Shutdown a Partition . . . . . 2
Remote Direct Memory Access (RDMA) Overview . . . . . 3
Network Table Services Overview . . . . . 3
SNI Filesets . . . . . 4

**Chapter 2. Configuring Switch Network Interface** . . . . . 5
Configuration Prerequisites . . . . . 5
Setting Up the NIM Server . . . . . 6
Configuring SNI and Multi-Link Interface . . . . . 10
Tuning SNI . . . . . 11

**Chapter 3. Error Messages** . . . . . 13
chgsni Error Messages . . . . . 13

**Chapter 4. SNI Commands** . . . . . 15
chgmlt Command . . . . . 15
chgsni Command . . . . . 16
defmlt Command . . . . . 18
ntblclean Command . . . . . 19
ntblstatus Command . . . . . 21

**Chapter 5. SNI Subroutines** . . . . . 23
ntbl_adapter_resources Subroutine . . . . . 23
ntbl_clean_window Subroutine . . . . . 24
ntbl_disable_window Subroutine . . . . . 26
ntbl_enable_window Subroutine . . . . . 27
ntbl_load_table Subroutine . . . . . 28
ntbl_load_table_rdma Subroutine . . . . . 29
ntbl_query_window Subroutine . . . . . 32
ntbl_status_adapter Subroutine . . . . . 33
ntbl_unload_window Subroutine . . . . . 34
ntbl_version Subroutine . . . . . 35

**Appendix. Notices** . . . . . 37
Trademarks . . . . . 38

**Index** . . . . . 39

# About This Book

This book provides information about concepts, tools, and techniques for using switch network interface (SNI) for eServer pSeries High Performance Switch (HPS). In addition, it includes the reference information for subroutines and commands specific to SNI. Use the information in this book if you plan to use an HPS or are managing an HPS on your system.

This edition applies to AIX Version 6.1 and to all subsequent releases of this product until otherwise indicated in new editions.

## Who Should Use This Book

This book is intended for network administrators, enterprise system administrators, experienced system administrators, system engineers, and system programmers who are configuring or managing one or more eServer pSeries High Performance Switches.

## Highlighting

The following highlighting conventions are used in this book:

| | |
|---|---|
| **Bold** | Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects. |
| *Italics* | Identifies parameters whose actual names or values are to be supplied by the user. |
| `Monospace` | Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type. |

## Case-Sensitivity in AIX

Everything in the AIX operating system is case-sensitive, which means that it distinguishes between uppercase and lowercase letters. For example, you can use the **ls** command to list files. If you type `LS`, the system responds that the command is ″not found.″ Likewise, **FILEA**, **FiLea**, and **filea** are three distinct file names, even if they reside in the same directory. To avoid causing undesirable actions to be performed, always ensure that you use the correct case.

Case-sensitive file names on AIX can also cause problems for personal computer clients running Windows operating systems because these operating systems normally treat file names as caseless. AIX file names that differ only in case would be perceived as the same file name from a PC client.

## ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

## Related Information

See the following books for additional information:

- *pSeries High Performance Switch Planning, Installation, and Service*
- *eServer Hardware Management Console Installation and Operations Guide*
- *RS/6000 SP System Performance Tuning Update*, an IBM Redbooks publication
- *Networks and communication management*
- *AIX Version 6.1 Commands Reference*

- *Performance management*
- *Installation and migration*

# Chapter 1. Overview of Switch Network Interface

This section contains information about the following topics:


- "Switch Network Interface and Multi-Link Support Overview"
- "User Space Job Support" on page 2
- "Technical Large Pages Overview" on page 2
- "Remote Direct Memory Access (RDMA) Overview" on page 3
- "Network Table Services Overview" on page 3
- "SNI Filesets" on page 4


## Switch Network Interface and Multi-Link Support Overview

Switch network interface (SNI) and multi-link support are the AIX components that provide support for the eServer pSeries High Performance Switch (HPS). The HPS is a network data transfer system using technology based on the architecture of SP Switch and SP Switch2. This technology is available for the IBM eServer p5 590/595 and 575 clustered servers. It can be used to help increase the communication bandwidth between servers and partitions within the cluster. Some additional benefits of implementing this communication subsystem include the following:

- Parallel, interconnected communication channels that form a unified switch network
- Opportunity to improve communication bandwidth and reductions in latency
- Option to use either fiber optic cables or copper cables for switch-to-switch network connections
- Improved reliability, availability, and serviceability (RAS)

SP Switch technology used several types of switch adapters to connect system components to the switch network. With the HPS, all network connections pass through an interface card packaged as either a 2-Link SNI or a 1-Link SNI. Each link on the SNI allows messages to pass between the server bus and the switch.

For example, using the SP Switch2, a Cluster 1600 switch network could be configured with two SP Switch2 adapters at each communication point. With the SP Switch2, the two switch adapters formed two independent switch networks. If one of the adapters developed a fault, the network experienced reduced performance.

In a similar manner, the HPS can also use SNIs with two or four links at each communication point. However, with the HPS, when one of the interfaces develops a fault, its load is automatically directed to the other links through the software. Although there might be reduced performance at the failed interface component, the overall performance of the HPS network remains unchanged.

For detailed information about the HPS, see *pSeries High Performance Switch Planning, Installation, and Service*, order number GA22-7951.

## SNI Support on AIX

Support for HPS on AIX consists of two logical device names: `sni`*x* and `mlt0`, where *x* is the device's minor number. The logical device `sni`*x* refers to the logical device name of one of the external links on a 2-link or 1-link SNI. The 2-linkSNI has multiple links, and AIX considers each external link a distinct device. For example, the 2-link SNI for HPS will present 2 devices, `sni0` and `sni1`. Each device will be administered independently. Attributes of the SNI can be changed using the **chgsni** command. For information about changing the attributes using the **chgsni** command, see "chgsni Command" on page 16.

The corresponding IP interface for each `sni`*x* logical device is `sn`*x*. The `sn`*x* IP network interface is configured like any other IP network interface, and it functions like any other IP network interface.

The multilink interface consists of the mlt0 logical device. Only one multilink support interface occurs per operating system instance. The related IP network interface for this device is ml0. The ml0 interface distributes all its network traffic over the switch network interfaces. The ml0 IP network interface is configured like any other IP network interface, and it functions like any other IP interface. Sending network traffic over the ml0 interface is not necessary, but it can help improve performance by distributing the traffic over the underlying snx interfaces. Sending network traffic over the ml0 interface also improves the Reliability and Serviceability characteristics. If one of the underlying snx interfaces suffers a fault, the ml0 interface automatically transfers its load to the other interfaces. When the fault is removed, normal operations resume automatically.

## User Space Job Support

The term *user space* means that communication on the link is done by the user application directly, without having to copy the message from or to the kernel. SNI supports the running of user space jobs. User space jobs are parallel applications started by a job scheduler using the network table subroutines. The supported message-passing APIs are Message Passing Interface (MPI) and the Low Level API (LAPI). Each SNI supports up to 64 *windows* for running parallel applications. A window is a virtual port or connection that allows applications to perform message passing over the HPS.

For more information about network table subroutines, see Chapter 5, “SNI Subroutines,” on page 23.

## Technical Large Pages Overview

SNI supports Technical Large Pages (TLP), which help provide high performance by decreasing the amount of address translation while copying, sending, or receiving large blocks of data. The SNI’s IP interface can use TLP for backing its send and receive pools. SNI also supports parallel applications that choose to use TLP for communications data areas. Such applications can send and receive data directly to and from TLP.

**Note:** If the required number of TLP is not available, the SNI devices do not configure.

Use the following equation to determine how large to make your TLP pool:

```
number of large pages = A + B + C + D
```

Where:

- A = 1 + (number_of_sni * 2)
- B = (number_of_sni * total_num_windows)
- C = (num_adpt * ((num_windows * 0x40000 + 0xFFFFFF) / 0x1000000));
- D = (send pool size + receive pool size) / 0x1000000
- total_num_windows = num_windows + 7

In the equation, the number_of_sni refers to the number of snix logical interfaces present in the partition. You can determine the num_windows, send pool size, and receive pool size for the AIX partition by using the following command:

```
lsattr -El snix
```

Where x is the device minor number. For example, 0, 1, 2 or 3.

For more information about using large pages, see Large Page Support in *Performance management*.

## Shutdown a Partition

Use the **shutdown** command to reboot a partition. This command can be issued from the AIX partition, the Hardware Management Console (HMC) GUI, or the Cluster Systems Management (CSM) server.

## Remote Direct Memory Access (RDMA) Overview

SNI support software contains RDMA capability. RDMA is the transport capability that allows processes running on one node to directly access (read or write against) the memory of processes running on a different node connected by an RDMA-capable network. The SNI adapters at both ends affect the transfer of data without any protocol processing on the target end point of the transport operation. The target end point is the one that does not initiate the transport.

RDMA can be used to:

- To remove processors from the movement of data. This allows for better overlap of computation and communication.
- To reduce the stress on the memory subsystems by reducing the number of bus crossings. There is one IO bus crossing instead of the traditional two memory buses and one IO bus crossing.
- To give users a one-sided programming model.
- To allow the protocols to efficiently stripe a message or different messages from a single task across multiple SNI to make use of the available communication bandwidth in parallel.
- To provide improved raw transport performance. If the transport bottleneck is the copy rate (memory bandwidth), the processor capacity, or the matching and early arrival buffering overhead, use of RDMA might provide better transport bandwidth.

## Using RDMA

By default, the RDMA capability of the SNI adapters is turned off. Use the **chgsni** command to turn on RDMA by setting the *rdma_xlat_limit* attribute for the SNI to a value greater than 0x0. 0x0 is the default value. The *rdma_xlat_limit* attribute controls the maximum amount of memory allowed for RDMA on an operating system instance (OSI).

**Note:** When RDMA is enabled by changing *rdma_xlat_limit* to a value greater than 0, it is possible for an RDMA-enabled job to exceed the AIX pin limit and to hang the system. If the pin limit is exceeded slowly enough the parallel application will catch a SIGDANGER signal and kill the job. However, if the pin limit is exceeded too rapidly, AIX may resort to killing processes, including systems processes, which may cause the system to appear to be hung. This situation may be prevented by setting the *rdma_xlat_limit* to a number that is smaller than 4096 times the sum of the number of large and small pages on the system.

When RDMA is turned on, the IP protocol automatically uses the RDMA capability. For user space jobs, an additional LoadLeveler keyword (*bulkxfer=yes)* must be set in the LoadLeveler job control file to indicate that this job is requesting RDMA. This ensures that users are able to run both RDMA and non-RDMA jobs on the system. For more information, please see the *LoadLeveler Using and Administration Guide*.

**Note:** The RDMA option must be turned on or off for the entire cluster. If this option is not set homogeneously across the entire cluster, SNI adapters cannot communicate because there are version mismatch failures.

## Network Table Services Overview

The network table services provide an application programming interface that is necessary to start applications that use the HPS. The network table describes all the information required for the tasks in a job to communicate with each other. The network table services provide a thread-safe shared library (with both 32-bit and 64-bit objects) and a set of commands to interact with the device's ioctl subroutines, which manipulate network tables and SNI device windows.

Job schedulers interact with the network table services in order to load, unload, and query network tables for parallel applications running over the HPS. The network table services also provide job schedulers with

SNI device information needed for scheduling. The new network table services are similar to the job switch resource table (JSRT) services provided for SP Switch systems.

For more information about network table subroutines, see Chapter 5, “SNI Subroutines,” on page 23. For more information about network table commands, see “ntblclean Command” on page 19 and “ntblstatus Command” on page 21.

## SNI Filesets

The following filesets are shipped with SNI:

*Table 1. SNI filesets and their descriptions*

| Fileset | Fileset description |
|---|---|
| **devices.common.IBM.sni.rte** | Switch Network Interface Runtime |
| **devices.common.IBM.sni.ml** | Multi Link Interface Runtime |
| **devices.common.IBM.sni.ntbl** | Network Table Runtime |
| **devices.chrp.IBM.HPS.rte** | IBM eServer pSeries High Performance Switch (HPS) Runtime |
| **devices.chrp.IBM.HPS.hpsfe** | IBM pSeries HPS Functional |
| **devices.msg.en_US.common.IBM.sni.rte** | Switch Network Interface Messages |
| **devices.msg.en_US.common.IBM.sni.ml** | Multi Link Interface Runtime |
| **devices.msg.en_US.common.IBM.sni.ntbl** | Network Table Runtime Messages |
| **devices.msg.en_US.chrp.IBM.HPS.rte** | pSeries HPS Runtime Messages |
| **devices.msg.en_US.chrp.IBM.HPS.hpsfe** | pSeries HPS Functional Utility Messages |

# Chapter 2. Configuring Switch Network Interface

This section contains information about preparing your system to use an HPS and configuring SNI. Certain hardware and software requirements must be met to be able to use an HPS. In addition, a NIM server may to be configured to perform the proper installation and initial configuration of the SNI and multilink interface. After configuration, AIX commands can be used to manage and tune the SNIs and multilink interfaces.

This sections contains the following topics:


- "Configuration Prerequisites"
- "Setting Up the NIM Server" on page 6
- "Configuring SNI and Multi-Link Interface" on page 10
- "Tuning SNI" on page 11


## Configuration Prerequisites

Before proceeding with the configuration, make sure that your system meets the following hardware and software requirements. For additional information, see *pSeries High Performance Switch Planning, Installation, and Service*, order number GA22-7951.

### Hardware Requirements

Clustered servers configured with the HPS must have the following hardware components:

- pSeries HPS
  - All HPSs that you will be using

    **Note:** It is not required to have one switch per frame. Some frames can be attached to a switch in another frame. However, only one switch can be attached to a single frame.
  - Additional switches must be mounted in a separate machine type 7040-W42 frame
- 2-Link or 1-Link switch network interface cards
  - 1-Link Switch Network Interface
    - Supported on IBM eServer p5 590 and IBM eServer p5 595 servers
    - Configured for fiber optic switch cables

    **Note:** This SNI feature must be ordered in pairs.
  - 2-Link Switch Network Interface
    - Non-interchangeable features designed for IBM eServer p5 575 servers.
    - All feature types configured for copper switch cables.
- IBM eServer p5 590 and 595 servers – 1 to 16 servers per pSeries HPS cluster with the required Hardware Management Consoles (HMC) and management server
- IBM eServer p5 575 server – 1 to 128 servers per pSeries HPS cluster with the required Hardware Management Consoles (HMC) and management server
- Switch cables
  - Server-to-switch: 1 m, 3 m or 10 m copper cables or 3.5 m, 10 m or 30 m fiber optic cables
  - Switch-to-switch: up to 40 m fiber-optic cables or 10 m copper cables (maximum lengths)
- HMC cables
  - RS-232 to the CEC service processors
  - RS-422 to frame power bulk power controller

- Administrative LAN Ethernet cables connecting all servers (including the management server) and all HMCs in the cluster

## Software Requirements

The following software is required:

- AIX 5.2 with 5200-06
- High Performance Switch Network Manager (HPSNM) – Switch management software (located on CSM MS)
- IBM Cluster Systems Management for AIX 5L (CSM)
- IBM Web-based System Manager GUI
  - Cluster-Ready Hardware Server
- Diagnostic software including:
  - Service Focal Point (SFP) – Problem management software (located on HMC)
  - Service Agent – Service Gateway function between IBM and customer
  - Inventory Scout Services – Vital Product Data and Microcode Management software

Optional AIX 5.2 with 5200-06 fileset:

- Reliable Scalable Cluster Technology (RSCT) – Components include:
  - HATS
  - HAGS
  - configRM
  - IBM Virtual Shared Disk
  - Low-level communications application (LAPI)

Recommended software packages:

- Cluster Systems Management for AIX 5L (CSM)
- Web-based System Manager
- General Parallel File System (GPFS)
- LoadLeveler for AIX 5L
- Parallel Environment (PE)

# Setting Up the NIM Server

Initial configuration of the HPS is performed through the NIM Network Installation Management server. This section describes the steps needed to configure NIM so that it can properly install the HPS environment and configure SNI and multi-link interface on the target LPAR nodes. This section assumes that you have knowledge of how to use NIM. For more detailed or introductory information about NIM, see *Network Installation* in *Installation and migration*.

## Update the NIM Server

Follow these steps to update your NIM server:

1. Install the AIX 5.2 with 5200-06 maintenance package.
2. Place required APARs and HPS filesets in the **AIX52I LPPSOURCE** directory using the **gencopy** command. For a list of filesets to copy, see “SNI Filesets” on page 4.
3. Create a new AIX52 NIM SPOT resource, or update your current AIX52 NIM SPOT resource after you have copied all required APARs and HPS filesets into your **AIX52I LPPSOURCE** directory..

## Create a Stanza File

After the NIM server has been updated, create a stanza file, which will include configuration information for the HPS. Each stanza begins with the name of the node and is followed by a series of lines in an *attribute=value* format. The stanza file will contain a stanza for each SNI device being configured.

Some of the attributes required in the stanza file can be gathered using the **getadapters** command. Other attributes, such as **netaddr** and **subnet_mask** must be entered manually. In addition, if there are `ml0` devices, the stanzas must be added manually. The **getadapters** command gathers all possible attributes, but the remaining required attributes must be added manually to the stanza file.

For a list of required attributes, see “Stanza Attributes” on page 8.

### Gathering Attributes Using the getadapters Command

The **getadapters** command can be used to gather some of the required information and to create an initial version of the stanza file. For example, running the following command would gather information from the `clstrn01` node and, with the data collected, create a stanza file named `mystanzafile`:

```
getadapters -z mystanzafile -n clstrn01
```

If a stanza file does not exist, the **getadapters** command will create one. If a stanza file with the specified name already exists when you run the **getadapters** command, the file will be updated with any additional information that is gathered from the nodes. Updates to the stanza file will be made according to the following guidelines. Consider these guidelines if you are planning to use the **getadapters** command to update your stanza file.

For Ethernet adapters, the following guidelines are used:

- The MAC address value is used to identify a unique Ethernet adapter stanza. This value is always returned for Ethernet adapters. The MAC address is unique across the cluster.
- If the stanza does not already exist for a particular MAC address, a new stanza will be created.
- If a stanza with the MAC address already exists and the location code is also the same as the newly gathered location code, a new stanza will not be created and the existing stanza will not be updated.
- If a stanza with the MAC address already exists but the location code is different from the newly gathered value, the existing stanza will be updated with the new location code.

For SNI devices, the following guidelines are used:

- The location code attribute value will be used to identify a unique SNI stanza. This value is always returned for SNI devices. The location code is only unique for a node in a Central Electronics Complex (CEC).
- If a stanza does not already exist for a particular location code, a new stanza will be created.
- If a stanza file already exists containing a particular location code, no new stanza will be created.

No stanza will be automatically removed from a stanza file when it is being updated. For example, if some adapters or SNI devices have been removed, either the stanza will have to be edited manually or a new stanza file will have to be created.

### Gathering Information Manually

Some of the attributes that are required for configuration cannot be gathered by the **getadapters** command and must be entered manually. These attributes include the IP address and subnet mask. For those attributes that must be added manually, gather the necessary values and add them to the stanza file according to the format given in “Stanza File Format.”

### Stanza File Format

The format of the stanza file must be consistent with the format required by the **nimadapters** command. Each stanza will contain the attributes and values needed by NIM, along with some additional attributes that will be used by CSM. After the stanza file has been completed, it will be used by the **getadapters**

command to store installation adapter information in the CSM database. It is also used by the **nimadapters** command to create adapter and SNI device configuration files and store them in the NIM adapter resource directory. Each command extracts the information it needs and ignores what it does not need.

The following are examples of an Ethernet adapter stanza, two SNI device stanzas, and a multi-link stanza. These examples include what would be created by the **getadapters** command in addition to other information that would have to be added manually.

```
clstrn01:
   machine_type=secondary
   network_type = en
   netaddr=10.10.1.4
   cable_type=N/A
   location=P1-I1/E2
   MAC_address=0003AC6E05E2
   subnet_mask=255.255.255.0
clstrn01:
   machine_type=secondary
   network_type=sn
   netaddr=192.169.0.4
   location=UP1.18-P1-H4/W3
   subnet_mask=255.255.255.0
clstrn01:
   machine_type=secondary
   network_type=sn
   netaddr=192.169.1.4
   location=UP1.18-P1-H4/W4
   subnet_mask=255.255.255.0
clstrn01:
   machine_type=secondary
   network_type = ml
   netaddr = 10.10.11.4
   subnet_mask=255.255.255.0
```

## Stanza Attributes

The following sections contain lists of the valid stanza attributes. The descriptions also indicate which ones will be created automatically by the **getadapters** command. In some cases, the attribute only is added to the initial stanza and in some cases, both the attribute and value are added. Additional attributes and values may be added manually as needed.

***Required NIM attributes:*** The following NIM attributes are required in the stanza file

**machine_type**
: Should be set to `secondary` for SNI devices that need to be configured and `install` for the install adapter. This attribute is always created and a value is always provided.

**network_type**
: The type of the network interface. The possible values are `en`, `et`, `sn`, and `ml`. This attribute is always created, but the value is only filled in if it can be determined by the **getadapters** command.

**netaddr**
: The IP address of the link or interface. This attribute is always created, but no value is provided.

**subnet_mask**
: The subnet mask used for the interface. This attribute is always created, but no value is provided.

**cable_type**
: Required if the **network_type** attribute is `en` or `et`. This attribute and default value will be added to the stanza.

***Required CSM attributes:*** The following CSM attributes are required in the stanza file.

**MAC_address**
This attribute is created and the value is provided if it can be determined by the **getadapters** command.

**ping_status**
Indicates if this SNI device can be used to ping the CSM management server. An `ok` value indicates that it can be used. If the **getadapters** command can determine a value, the attribute and value will be added.

**adapter_duplex**
The duplex value of the SNI device. If the value is provided on the **getadapters** command line, it will be added to any stanza with a **ping_status** of `ok`.

**install_server**
The machine that will be used to install the node. If the value is provided on the **getadapters** command line, it will be added to any stanza with a **ping_status** of `ok`.

**install_gateway**
The gateway to use for the node to contact the management server. If the value is provided on the **getadapters** command line, it will be added to any stanza with a **ping_status** of `ok`.

**adapter_speed**
The SNI device speed. If the value is provided on the **getadapters** command line, it will be added to any stanza with a **ping_status** of `ok`.

***Optional NIM Attributes:*** The following attributes are optional.

**location**
Physical location of this SNI device, or location code. This attribute is always created, and the value is provided if it can be determined by the **getadapters** command.

**interface_name**
The name of the network interface. If the **getadapters** can determine a value, the attribute and value will be added.

**attributes**
Blank-separated list of additional network attributes. This attribute will not be added to the initial stanza.

**secondary_hostname**
The hostname to be saved in the **/etc/hosts** directory along with the **netaddr** value. This attribute will not be added to the stanza.

**media_speed**
Optional if **network_type** is `en` or `et`. This attribute will not be added to the stanza.

**comments**
Optional descriptive text. This attribute will not be added to the stanza.

## Create a NIM adapter_def resource

The NIM **adapter_def** resource represents a directory that contains SNI device configuration files that may be used during a NIM **bos_inst** or **cust** operation. These configuration files are created when you use the **nimadapters** command with the stanza file that was created previously. To create the **adapter_def** resource, you must select a directory location and use the NIM **define** operation.

For example, running the following command will create the **/export/nim/adapt_defs** directory for storing configuration files and also create an **adapter_def** resource called `my_adapter_res`:

```
nim -o define -t adapter_def -a server=master -a location=/export/nim/adapt_defs my_adapter_res
```

## Run the nimadapters command

The **nimadapters** command takes the information from your stanza file and creates SNI device configuration files in the directory you specified when you created the **adapter_def** resource.

For example, using the stanza file called **mystanzafile** and the **adapter_def** resource called my_adapter_res, you could issue the following command.

```
nimadapters -d -f mystanzafile my_adapter_res
```

This will create a configuration file for each node mentioned in the stanza file. The files will be created in the **/export/nim/adapt_defs** directory.

### Configure the adapters

To configure the additional SNI devices during the initial installation of the node, you may include the **adapter_def** resource on the command line when you issue the NIM **bos_inst** operation. This operation is one of the last steps in the installation process described in the CSM Planning and Installation guide. Refer to this documentation for more information on the installation process. The SNI device configuration tasks described previously should be completed after the CSM management server has been installed, the nodes have been defined, and the NIM objects and resources have been defined.

The following example assumes an rte type installation, using the resources contained in the resource group **node_res_grp**. Because this is an initial installation of the node, the **boot_client** value is set to no. In this case the network boot of the node must be initiated manually or by using the CSM **netboot** command:

```
nim -o bos_inst -a source=rte -a adapter_def=my_adapter_res -a group=node_res_grp -a accept_licenses=yes -a boot_client=no clstrn01
```

**Note:** The **my_adapter_res** resource could also have been added to the **node_res_grp** resource group before running this command, in which case it would not have to be included on the command line.

If the nodes are already installed you can use the NIM **cust** operation to configure the SNI devices. For example, using the **my_adapter_res** resource, you could issue the following command to configure the SNI devices on node clstrn01:

```
nim -o cust -a adapter_def=my_adapter_res clstrn01
```

After the **cust** operation has completed, the node must be rebooted in order for the configuration to take effect.

---

## Configuring SNI and Multi-Link Interface

Follow these instructions to configure SNI and add an IP address to the multi-link interface. In this procedure, SNI must be configured before the multi-link interface, which allows IP packets to be distributed across the SNI interfaces.

## Subnet Considerations

When configuring SNI and the multi-link interface, the multi-link interface needs to be configured on a different subnet from any of the switch network interfaces. The switch network interfaces can be configured on the same subnet or on separate subnets, but they cannot be configured on the same subnet as the multi-link interface.

Because an operating system instance can have multiple SNIs, it is good practice to put them on as many subnets as possible. Having as many different subnets as operating-system instances allows the Reliable Scalable Cluster Technology (RSCT) peer domains to more accurately detect the availability of SNIs.

## Configure SNI

To configure an SNI interface, follow these steps:

1. At the command line, type `smit chsn`.
2. Select the interface you want to configure.
3. Enter values for the attributes. Press Enter.
4. At the command line, type `smit mtu`.

5. Change the MTU size to the desired value.

## Configure Multi-link interface

To configure a multi-link interface, follow these steps.

1. At the command line, type `smit ibm-ml`.
2. Select **Change/Show Characteristics of a Multi-Link Device**.

   **Note:** The multi-link interface must be configured on a different subnet from any of the SNIs.
3. Change the desired attributes.
4. Select **Change/Show Characteristics of a Multi-Link Interface**.
5. Change the desired attributes and press Enter.

# Tuning SNI

The following are SNI tuning recommendations.

## Number of Windows

The SNI device support code was restructured to deliver higher communication performance in terms of higher bandwidth and lower latency. Most of the restructuring is only surfaced through better performance but some changes do have implications for the administrator.

The number of MPI and LAPI windows is now a tunable parameter with a default of 16 windows and a maximum of 64 windows. To display the current settings use the following command:

```
lsattr -El snix
```

Where `x` is the device minor number. For example, 0, 1, 2, or 3.

To change this setting, see “chgsni Command” on page 16.

**Note:** During the SNI device configuration, the SNI device driver uses this setting to calculate the number of Technical Large Pages (TLP) needed. If the required number of TLP is not available, the SNI devices do not configure.

The size of the buffers allocated by the SNI device driver starts at 4096 bytes, and increases to 65536 bytes in values of 2.

If the size of the data being sent is just slightly larger than 4 K, 8 K, 16 K, 32 K, or 64 K, for example, the buffer allocated from the pool is the next larger size. This can cause as low as 50 percent efficiency in usage of the buffer pools because more than half of the pool can go unused in certain circumstances.

When assembling packets, there is always one mbuf from the IP mbuf pool used to assemble the packet header information in addition to any data buffers from the spool. If the mbuf pool size is too small, and the system runs out of mbufs, the packet is dropped. The mbuf pool is used globally for all IP traffic, and is set using **thewall** tunable with the **no** command. For more information about the **no** command, see *AIX Version 6.1 Commands Reference*.

When you are sending 4 KB of data over the switch, an mbuf from the mbuf pool will be used, as well as one 4 KB spool buffer for the data. If the amount of data being sent is less than 200 bytes, no buffer from the spool is allocated, because there is space in the mbuf used for assembling the headers to stage the data. However, if sending 256 bytes of data, one mbuf for the IP headers and one 4 KB send pool buffer for the data, will be used. In this situation, 15/16 of the buffer space in the send pool is wasted. These same scenarios apply to the receive pool when a packet is received on a node.

The key for peak efficiency of the spool and rpool buffers is to send messages that are at or just below the buffer allocation sizes, or less than 200 bytes.

When tuning the rpool and spool, it is important to know the expected network traffic. If the size of the buffers for the applications is not optimum, much of the spool and rpool are wasted. This inefficient usage requires the need to increase the rpool and spool sizes. When allocating the rpool and spool, realize that this space is pinned kernel space in physical memory. This takes space away from user applications and is particularly important.

If there is a small number of active sockets, there is usually enough rpool and spool space that can be allocated. A system in which a node has a large number of sockets opened can easily run out of spool space when all the sockets transmit at once.

On the receive side of a parallel or client/server implementation, where one node acts as a collector for several other nodes, the rpool has the same problem. Four nodes, each with 600 sockets, each sending to one node two 1 KB packets, will exceed the rpool limit, but those same sockets, each sending twice as much data, 4 KB in one 4 KB packet, will work correctly. The solution is sending a single larger packet rather than several smaller ones.

# Chapter 3. Error Messages

This section contains the following error message information:

- "chgsni Error Messages"

## chgsni Error Messages

*Table 2. Message numbers, message text, and recovery information.*

| Message Number | Message Text | Recovery Information |
|---|---|---|
| 2506-100 | No attributes requested to change! | Request changes to one or more attributes. |
| 2506-101 | Attributes out of range! | Ensure the attributes requested are in-range. |
| 2506-102 | Error during odm_initialize | Try again later or contact system administrator about insufficient storage. |
| 2506-103 | Error during odm_lock | Try again later if the ODM is locked by another process, or contact system administrator about insufficient storage. |
| 2506-104 | get_odm_vals failed with ret =%d | Ensure specified attributes are valid. |
| 2506-105 | change_driver failed with ret = %d | Ensure specified logical name is valid. |
| 2506-106 | update_odm failed with ret = %d | Ensure specified attributes are valid. |
| 2506-107 | No spaces found in -a arg. Exiting | Supply the **-a** option with the attributes you want modified. |
| 2506-108 | Badly formed -a parameter. | Follow correct command option syntax. |
| 2506-109 | malloc failed! Returning E_MALLOC | Contact system administrator about insufficient storage. |
| 2506-116 | CuDv odm_get_first(CuDv) failed with odmerrno %d | Ensure specified attributes are valid. |
| 2506-117 | Invalid attribute: %s | Ensure specified attributes are valid. |
| 2506-118 | Failed to open device %s | Ensure specified logical name is valid. |
| 2506-119 | ioctl to change %s failed with errno = %d | Ensure specified logical name or attribute and attribute values are valid. |
| 2506-120 | Failed to open device %s | Ensure specified logical name is valid. |
| 2506-121 | (%s) ioctl %d failed with errno = %d | Ensure specified logical name or attribute/attribute values are valid. |
| 2506-122 | odm_add_obj for %s failed with odmerrno = %d | Contact your ODM administrator. |
| 2506-123 | Failed to terminate the ODM session with odmerrno = %d | Contact your ODM administrator. |

*Table 2. Message numbers, message text, and recovery information. (continued)*

| Message Number | Message Text | Recovery Information |
|---|---|---|
| 2506-124 | Failed to execute the **savebase** command in a subshell with errno = %d. Any updated attribute will be lost after rebooting the system. You should try to run the **/usr/sbin/savebase** command. | Manually run **/usr/sbin/savebase** command. |
| 2560-125 | No device name supplied. | Specify a valid device name. |
| 2560-126 | Invalid num_windows, val = %lld | Specify a number between 1 and 64. |

# Chapter 4. SNI Commands

This section contains reference information for the following SNI commands:


- "chgmlt Command"
- "chgsni Command" on page 16
- "defmlt Command" on page 18
- "ntblclean Command" on page 19
- "ntblstatus Command" on page 21


## chgmlt Command

### Purpose

Changes the attributes of a multi-link device.

### Syntax

**chgmlt -l** *DeviceName* { **-a** *Attribute*=*Value* } [ **-a** *Attribute*=*Value* ] . . .

### Description

The **chgmlt** command changes one or more attributes of the Switch Network Interface's multi-link driver.

### Flags

**-a** *Attribute*=*Value*

Changes the value of the specified device attribute. *Attribute* can be one of the following:

**agg_list=***List*
: The list of native network interfaces of type Switch Networks (SN) that the multi-link device can use. If the **agg_list** is not specified, the multi-link device will create an **agg_list** of all network interfaces of type Switch Network that are defined each time the multi-link driver is configured.

**agg_interval=***Seconds*
: The number of seconds between connectivity probes.

**agg_threshold=***Threshold*
: The number of connectivity probes before route times out.

**-l** *DeviceName*

Specifies the device logical name in the Customized Devices object class whose attribute values should be changed.

### Exit Status

**0** The command completed successfully.

**>0** An error occurred.

### Security

You must have root authority to run this command.

## Examples

1. To change the switch network interfaces that are included in the multi-link interface, type:

```
chgmlt -l mlt0 -a agg_list="sn0,sn1,sn2,sn3,sn4"
```

## Location

**/etc/methods/chgmlt**

## Standard Error

This command writes error messages to standard error.

## Related Information

The **lsattr** command.

---

# chgsni Command

## Purpose

Applies configuration changes to a Switch Network Interface device.

## Syntax

**chgsni -l** *Name* **-a** '*Attribute*=*Value* [ *Attribute*=*Value* ]' [ **-v** ]

## Description

The **chgsni** command changes the device memory or window resource allocations for the Switch Network Interface device.

The Switch Network Interface device multiplexes between independent data streams, where a data stream is represented by an SNI device window. A subset of SNI device windows are for system use only. For example, the IP window. Windows that are neither held for system use nor otherwise reserved may be allocated dynamically for large-scale parallel applications.

## Flags

**-a** *Attribute=Value*

Specifies the device attribute pairs used for changing specific attribute values. The *Attribute=Value* parameter can use one attribute value pair or multiple attribute value pairs for one **-a** flag. If you use a **-a** flag with multiple attribute value pairs, the list of pairs must be enclosed in quotation marks with spaces between the pairs. For example, entering **-a** *Attribute=Value* lists one attribute value pair per flag, while entering **-a** '*Attribute1=Value1 Attribute2=Value2*' lists more than one attribute value pair.

**num_windows=***number*

The number of user space windows available for MPI and LAPI applications.

Default value: 16

Minimum value: 1

Maximum value: 64

**rpoolsize=***size*

Size of the IP receive buffer pool (in bytes). This attribute is modified for all existing devices. Any user-supplied **-l** option is ignored.

Default value: 67108864 (64 megabytes)

Minimum value: 16777216 (16 megabytes)

Maximum value: 1073741824 (1 gigabyte)

**spoolsize=***size*

Size of the IP send buffer pool (in bytes). This attribute is modified for all existing devices. Any user-supplied **-l** option is ignored.

Default value: 67108864 (64 megabytes)

Minimum value: 16777216 (16 megabytes)

Maximum value: 1073741824 (1 gigabyte)

**rdma_xlat_limit=size**

Maximum amount of memory allowed for RDMA (in bytes). This attribute is modified for all existing devices. Any user supplied -l option is ignored.

Default value: 0x0 (Implies RDMA is turned off)

Minimum value: 0x0

Maximum value: 0xFFFFFFFFFFFFFFFF

Recommended value: 0x8000000000000000

**Note:** Device memory attribute sizes can be specified in either decimal or hexadecimal. Hexadecimal values must be preceded by ″0x″ or ″0X″.

**-l**

Specifies the device logical name in the Customized Devices object class whose attribute values should be changed.

**-v**

Specifies the verbose output option.

## Standard Output

This command writes informational messages to standard output whenever a device memory attribute is changed.

## Standard Error

This command writes error messages to standard error.

## Exit Status

| | |
|---|---|
| **0** | The command completed successfully. |
| **>0** | An error occurred. |

## Security

You must have root authority to run this command.

## Examples

1. To change the number of windows to 32, type:

   ```
   chgsni -l sni0 -a num_windows=32
   ```

2. To change the size of the IP send and receive buffers to 1 megabyte, type:

   ```
   chgsni -a 'rpoolsize=1048576 spoolsize=1048576'
   ```

## Location

**/etc/methods/chgsni**

## Related Information

The **lsattr** command.

The “chgsni Error Messages” on page 13.

---

# defmlt Command

## Purpose

Defines the multi-link device and its attributes.

## Syntax

**defmlt -l** *DeviceName* { **-a** *Attribute=Value* } [ **-a** *Attribute=Value* ] . . .

## Description

The **defmlt** command defines the Switch Network (SN) multi-link driver and its attributes.

## Flags

**-a** *Attribute=Value*
: Identifies the device attribute to be changed and the value to which it should be changed. *Attribute* can be one of the following:

  **agg_list=***List*
  : The list of native network interfaces of type Switch Networks (SN) that the multi-link device can use. If the **agg_list** is not specified, the multi-link device will create an **agg_list** of all network interfaces of type Switch Network that are defined each time the multi-link driver is configured.

  **agg_interval=***Seconds*
  : The number of seconds between connectivity probes.

  **agg_threshold=***Threshold*
  : The number of connectivity probes before route times out.

**-l** *DeviceName*
: Specifies the device logical name in the Customized Devices object class whose attribute values should be changed.

## Exit Status

**0**
: The command completed successfully.

**>0**
: An error occurred.

## Security

You must have root authority to run this command.

## Examples

1. To define the multi-link driver, type:

```
defmlt -l mlio -a 'agglist="sn0,sn1,sn1,sn3,sn4" agginterval=3 aggthreshold=10'
```

## Location

**/etc/methods/defmlt**

## Standard Error

This command writes error messages to standard error.

## Related Information

The **lsattr** command.

---

# ntblclean Command

## Purpose

Forces the unload of the network table window for a specified SNI device.

## Syntax

**ntblclean -h** | **-?** | [ [ **-k** ] **-w** *WindowID* **-a** *Adapter* ]

## Description

The **ntblclean** command overrides the job key checking and unloads the network table window on the SNI device specified. Normal unloading of the network table window by the **ntbl_unload_window** API checks that the **job_key** of the unload matches the **job_key** specified during the load. The **ntblclean** command ignores this check and allows the administrator to unload the window from a node. Use this command for error recovery and not for normal unloading. Use this command when a parallel job has left a process in use and the window failed to unload with the **ntbl_unload_window** API.

The *WindowID* parameter must be a window found in the list of available window IDs returned from the **ntbl_adapter_resources** API.

If the **-k** flag is not specified, the unload of the window will only be processed when the window is in the **NTBL_LOADED_STATE** or **NTBL_DISABLED_STATE**.

If the **-k** flag is specified, and the window is in the **NTBL_ACTIVE_STATE** or **NTBL_BUSY_STATE** state, the process using the window will be sent a **SIGKILL** signal. It is not guaranteed that the process will receive the signal. The unload of the window will be processed without the **SIGKILL** signal when the window is in the **NTBL_LOADED_STATE** or **NTBL_DISABLED_STATE** state.

A single job's network table may contain more than one window. Run this command for every window within the table. Use the **ntblstatus** command to obtain the current state of the windows.

## Flags

| | |
|---|---|
| **-?** | Displays the usage statement for the command. |
| **-a** *Adapter* | Specifies the SNI device device name upon which the window to be unloaded resides. |
| **-h** | Displays a description of each flag. |
| **-k** | Sends the process that is currently using the network table window a **SIGKILL** signal and unloads the window. The **-k** flag has the same function as the **ALWAYS_KILL** option of the **ntbl_clean_window** API. |
| **-w** *WindowID* | Specifies the window ID for which the unload will be done. |

## Exit Status

| | |
|---|---|
| **0** | The command completed successfully. |
| **>0** | An error occurred. |

## Security

You must have root authority to run this command.

## Standard Output

After the network table window is successfully unloaded, the status should be **NTBL_UNLOADED_STATE**.

## Related Information

The "ntblstatus Command" on page 21.

The “ntbl_adapter_resources Subroutine” on page 23, the “ntbl_clean_window Subroutine” on page 24, and the “ntbl_unload_window Subroutine” on page 34.

# ntblstatus Command

## Purpose

Displays the current status of a network table window. This command may display a single window, all windows on an SNI device or, all windows on all SNI devices.

## Syntax

**ntblstatus** [ **-h** | **-?** | **-w** *WindowID* **-a** *Adapter* | **-a** *Adapter* ]

## Description

The **ntblstatus** command reports the current status of a network table window. If the **-w** and **-a** flags are specified, status will be returned for a single window on the SNI device specified. If the **-a** flag is specified without a **-w** flag, status will be returned for all network windows on that SNI device. If no **-a** flag is specified, status is returned for all network windows on all switch SNI devices on the machine where the command is invoked.

## Flags

| | |
|---|---|
| **-?** | Displays the usage statement for the command. |
| **-a** *Adapter* | Specifies the SNI device name for which status of all windows will be returned. If used with the **-w** flag, status is returned only for the specified window on the SNI device. |
| **-h** | Displays a description of each flag. |
| **-w** *WindowID* | Specifies the window ID for which status will be returned. This flag requires the **-a** flag to specify the SNI device. |

## Exit Status

| | |
|---|---|
| **0** | The command completed successfully. |
| **>0** | An error occurred. |

## Security

Any user can run the **ntblstatus** command.

## Examples

1. To display the status of window 1 on the SNI device sni0, type:

   ```
   ntblstatus -w 1 -a sni0
   ```

   Output similar to the following is displayed window when the window’s state is NTBL_LOADED_STATE, NTBL_DISABLED_STATE, NTBL_BUSY_STATE or NTBL_ACTIVE_STATE.

   ```
   ************************************************************
   User: loadl Uid: 1064 Pid: 45228
   Job Description: c97n01.ppd.pok.ibm.com.8.0
   Time of request: Tue_Dec_11_17:27:52_EST_2001
   Memory Requested: 1048576 Memory Allocated: 1048576
   Adapter: sni0 Window id: 1 State: NTBL_LOADED_STATE
   ```

2. To display the status of window 5 on SNI device sni0, type:

   ```
   ntblstatus -w 5 -a sni0
   ```

Output similar to the following is displayed per window when a window's state is **NTBL_UNLOADED_STATE** or an error occurred:

```
************************************************************
Adapter sni0 window 5 returned NTBL_UNLOADED_STATE.
```

## Standard Output

The output of this command reports the following data when a network table window is loaded, disabled, active or busy:

**User** User name corresponding to the user ID.

**Job Description**
String specified upon load request for the job using the network table.

**Time of request**
Timestamp of the time that the load request was processed.

**Window id**
Window for which the data is being reported.

**Pid** ID of the process which issued the load request, most likely a job management application.

**Uid** User ID loaded onto the network table.

**Memory Requested**
Window memory requested for use by the task per window.

**Memory Allocated**
Window memory allocated for use by the task per window.

**State** Current state of the window.

**Adapter**
SNI device for which the status is being reported.

## Related Information

The "ntblclean Command" on page 19.

# Chapter 5. SNI Subroutines

This section contains reference information for the following SNI subroutines:


- "ntbl_adapter_resources Subroutine"
- "ntbl_clean_window Subroutine" on page 24
- "ntbl_disable_window Subroutine" on page 26
- "ntbl_enable_window Subroutine" on page 27
- "ntbl_load_table Subroutine" on page 28
- "ntbl_load_table_rdma Subroutine" on page 29
- "ntbl_query_window Subroutine" on page 32
- "ntbl_status_adapter Subroutine" on page 33
- "ntbl_unload_window Subroutine" on page 34
- "ntbl_version Subroutine" on page 35


## ntbl_adapter_resources Subroutine

### Purpose

Returns the resources of a specified SNI device to the node from which it is invoked.

### Library

Network Table Library (**libntbl.a**)

### Syntax

```
#include <ntbl.h>

int ntbl_adapter_resources (version,adapter,resources)
int version;
char *adapter;
struct ADAPTER_RESOURCES *resources;
```

### Description

The **ntbl_adapter_resources** subroutine obtains the configured resources of an SNI device specified on the node from which it is invoked. The **ADAPTER_RESOURCES** structure contains the following:

- Number of windows available on the SNI device for job scheduling
- List of window IDs available on the SNI device for job scheduling
- Maximum and minimum DMA memory available for request per window
- Aggregate amount of device memory available for all windows to request
- FIFO slot size
- Device type of the SNI device
- Logical ID of the SNI device
- Network ID of the SNI device.

All memory data is represented in bytes. Memory is allocated dynamically by the **ntbl_adapter_resources** subroutine for the *window_list* member of the **ADAPTER_RESOURCES** structure based on the *window_count* member's size. The caller is responsible for releasing the memory allocated for the *window_list* member using the **free** subroutine for each entry in the array.

The **ntbl_adapter_resources** subroutine can be used to obtain information needed by the **ntbl_load_table** subroutine.

## Parameters

| | |
|---|---|
| *version* | Specifies the version of header file used. This value is the **NTBL_VERSION**, which is defined in the **ntbl.h** file. |
| *adapter* | Specifies the SNI device name for which the configured resources are obtained. |
| *resources* | Specifies the address of the **ADAPTER_RESOURCES** structure that contains the configured resources. |

## Return Values

Upon successful completion, the **ntbl_adapter_resources** subroutine returns a value of **NTBL_SUCCESS**.

The following data members are returned within the **ADAPTER_RESOURCES** structure when **NTBL_SUCCESS** is returned:

| | |
|---|---|
| **unsigned int** *device_type* | Integer containing an LED value that represents the device type. |
| **unsigned short** *lid* | Logical ID of the SNI device. |
| **unsigned short** *network_id* | Network the SNI device resides on. |
| **unsigned long long** *max_window_memory* | Maximum memory allowed per window. |
| **unsigned long long** *min_window_memory* | Guaranteed minimum memory per window. |
| **unsigned long long** *avail_adapter_memory* | Aggregate amount of DMA device memory available for windows. |
| **unsigned short *** *window_list* | Array of window IDs that are available for use. |
| **unsigned int** *window_count* | Count of the windows. Size of the *window_list* member. |
| **unsigned long long** *fifo_slot_size* | Size of the FIFO slot. Used by the LoadLeveler to calculate **job_max**. |

If unsuccessful, the **ntbl_adapter_resources** subroutine returns an integer value defined in the **NTBL_RC** enumerator found in the **ntbl.h** file.

| | |
|---|---|
| **NTBL_SUCCESS** | Resources were obtained. |
| **NTBL_EINVAL** | Invalid input parameter. |
| **NTBL_EADAPTER** | Open failed on the *adapter* parameter. |
| **NTBL_EIOCTL** | The **ioctl** call failed. |
| **NTBL_EMEM** | Failed to allocate memory. |
| **NTBL_EIO** | SNI device reported down state. |

# ntbl_clean_window Subroutine

## Purpose

Overrides the job-key checking of the **ntbl_unload_window** subroutine and unloads the network table window on the node from which it is invoked.

## Library

Network Table Library (**libntbl.a**)

## Syntax

```
#include <ntbl.h>

int ntbl_clean_window (version, adapter, option, window_id)
int version;
char *adapter;
enum CLEAN_OPTION option;
unsigned short window_id;
```

## Description

The **ntbl_clean_window** subroutine overrides the job-key checking, sends a **SIGKILL** signal to the process using the window, and unloads the network table window on the node from which it is invoked.

Normal unloading of the network table window by the **ntbl_unload_window** subroutine verifies that the job key of the unload matches the job key specified during the load. The **ntbl_clean_window** subroutine ignores this verification and allows the caller to unload the window from a node. Use this subroutine for error recovery and not for normal network table unloading. For example, when a parallel job has left a process in use and the window failed to unload by the **ntbl_unload_window** subroutine, the caller can use the **ntbl_clean_window** subroutine to clean up that node. This subroutine should be called even when the SNI device reports **NTBL_EIO**. The resources held by the window will be released regardless of the state of the SNI device.

The *window_id* parameter must be a window found in the list of available window IDs returned from the **ntbl_adapter_resources** subroutine. If the **ntbl_adapter_resources** subroutine is reporting an **NTBL_EIO** state, window checking will be bypassed and the unload will be attempted.

If the **LEAVE_INUSE** option is specified, the unload of the window will only be processed when the window is in the **NTBL_LOADED_STATE** state or **NTBL_DISABLED_STATE** state.

If the **ALWAYS_KILL** option is specified, and the window is in the **NTBL_ACTIVE_STATE** state or the **NTBL_BUSY_STATE** state, the process using the window will be sent a **SIGKILL** signal, but the process might not receive the signal. The unload of the window will be processed without the **SIGKILL** signal when the window is in the **NTBL_LOADED_STATE** state or **NTBL_DISABLED_STATE** state.

You must have root user authority to invoke the **ntbl_clean_window** subroutine.

## Parameters

*version* Specifies the version of header file used. This value should be **NTBL_VERSION**, which is defined in the **ntbl.h** file.

*adapter* Specifies the SNI device device name to be opened. This is where the window resides.

*option* Contains the **CLEAN_OPTION** enumeration value. This value indicates the following action:

**LEAVE_INUSE**
If a process is currently using the network table window, do not send a signal to the process or try to unload the window. If no process is using the window, issue an unload.

**ALWAYS_KILL**
If a process is currently using the network table window, this option sends a **SIGKILL** signal to the process and unloads the window.

*window_id* Specifies the window ID for the window that will be unloaded.

## Return Values

Upon successful completion, the **ntbl_clean_window** subroutine returns a value of **NTBL_SUCCESS**.

Otherwise, it returns an error value defined by the **NTBL_RC** enumerator found in the **ntbl.h** header file.

| | |
|---|---|
| **NTBL_SUCCESS** | Window was successfully unloaded. |
| **NTBL_EPERM** | Caller does not have root authority. |
| **NTBL_EINVAL** | Invalid input parameter. |
| **NTBL_EADAPTER** | Open failed on the *adapter* parameter. |
| **NTBL_EIOCTL** | The **ioctl** call failed. |
| **NTBL_BUSY_STATE** | Window within the table was in use. |
| **NTBL_UNLOADED_STATE** | Window is not loaded. |
| **NTBL_ACTIVE_STATE** | Window is active. |
| **NTBL_EMEM** | Failed to allocate memory. |

---

# ntbl_disable_window Subroutine

## Purpose

Disables the specified window on the specified SNI device.

## Library

Network Table Library (**libntbl.a**)

## Syntax

```
#include <ntbl.h>

int ntbl_disable_window (version, adapter, window_id)
int version;
char *adapter;
unsigned short window_id;
```

## Description

The **ntbl_disable_window** subroutine disables a network table window for the specified SNI device on the node from which it is invoked. The caller of this subroutine must check the state of the window with the **ntbl_query_window** subroutine prior to issuing the **ntbl_disable_window** subroutine. The window must be in the **NTBL_ACTIVE_STATE** state or **NTBL_LOADED_STATE** state. If the **ntbl_disable_window** subroutine returns successfully, the state of the window will be changed to the **NTBL_DISABLED_STATE** state. The requested state change is not atomic. Use the **ntbl_query_window** subroutine to verify the change has occurred.

You must have root user authority to invoke this subroutine.

## Parameters

| | |
|---|---|
| *version* | Specifies the version of header file used. This value should be **NTBL_VERSION**, which is defined in the **ntbl.h** file. |
| *adapter* | Specifies the SNI device device name to be opened. This is where the window resides. |
| *window_id* | Specifies the window ID of the window that will be disabled. |

## Return Values

Upon successful completion, the **ntbl_disable_window** subroutine returns a value of **NTBL_SUCCESS**.

Otherwise, it returns one of the following error values defined by the **NTBL_RC** enumerator found in the **ntbl.h** header file:

| | |
|---|---|
| **NTBL_SUCCESS** | Disable was successful. |
| **NTBL_EPERM** | Caller did not have root user authority. |
| **NTBL_EINVAL** | Invalid input parameter. |
| **NTBL_EADAPTER** | Open failed on the *adapter* parameter. |
| **NTBL_EIOCTL** | The ioctl call failed. |
| **NTBL_EIO** | SNI device reported down state. |
| **NTBL_BUSY_STATE** | The ioctl indicated the window was not in the correct state. |

## ntbl_enable_window Subroutine

### Purpose

Enables the specified window on the specified SNI device.

### Library

Network Table Library (**libntbl.a**)

### Syntax

```
#include <ntbl.h>

int ntbl_enable_window (version, adapter, window_id)
int version;
char *adapter;
unsigned short window_id;
```

### Description

The **ntbl_enable_window** subroutine enables a network table window for the specified SNI device on the node from which it is invoked. The state of the window must be **NTBL_DISABLED_STATE** in order to enable it. The state of the window will be changed from the **NTBL_DISABLED_STATE** state to the **NTBL_LOADED_STATE** state. The requested state change is not atomic. Use the **ntbl_query_window** subroutine to verify the change has occurred.

You must have root user authority to invoke this subroutine.

### Parameters

| | |
|---|---|
| *version* | Specifies the version of header file used. This value should be **NTBL_VERSION**, which is defined in the **ntbl.h** file. |
| *adapter* | Specifies the SNI device device name to be opened. This is where the window resides. |
| *window_id* | Specifies the window ID of the window that will be enabled. |

### Return Values

Upon successful completion, the **ntbl_enable_window** subroutine returns a value of **NTBL_SUCCESS**.

Otherwise, it returns an error value defined by the **NTBL_RC** enumerator found in the **ntbl.h** header file.

| | |
|---|---|
| **NTBL_SUCCESS** | Enable was successful. |
| **NTBL_EPERM** | Caller does not have root user authority. |
| **NTBL_EINVAL** | Invalid input parameter. |
| **NTBL_EADAPTER** | Open failed on the *adapter* parameter. |
| **NTBL_EIOCTL** | The ioctl call failed. |
| **NTBL_EIO** | SNI device reported down state. |
| **NTBL_BUSY_STATE** | The ioctl indicated the window is not in correct state. |

# ntbl_load_table Subroutine

## Purpose

Loads a network table on the node from which it is invoked.

## Library

Network Table Library (**libntbl.a**)

## Syntax

```
#include <ntbl.h>

int ntbl_load_table (version, adapter, network_id, uid, pid, job_key, job_desc, window_memory, table_size, table)
int version;
char *adapter;
unsigned short network_id;
uid_t uid;
pid_t pid;
unsigned short job_key;
char *job_desc;
unsigned long long *window_memory;
int table_size;
NTBL **table;
```

## Description

The **ntbl_load_table** subroutine loads the network table on the node from which it is invoked. This network table is used by parallel jobs running user space over the switch.

An **NTBL** structure must be defined for every task in the parallel job. Each structure defines the task, window, and SNI device relationship. The array of structures defines how the tasks of the parallel job will communicate. The windows specified must be valid and in the **NTBL_UNLOADED_STATE** state.

Valid *window_memory* ranges can be obtained from the **ntbl_adapter_resources** subroutine. The *window_memory* requested is not guaranteed to be satisfied by the device driver, thus this parameter will record the allocated memory upon return. A *window_memory* input value of 0 indicates that the device driver will determine the window memory size. This value will be the value of the **win_min_size** member set by the administrator using the **chgsni** command.

You must have root user authority to invoke this subroutine.

## Parameters

| | |
|---|---|
| *version* | Specifies the version of header file used. This value should be **NTBL_VERSION**, which is defined in the **ntbl.h** header file. |
| *adapter* | Specifies the SNI device name to be opened. |
| *network_id* | Specifies the network ID for the SNI device. |
| *uid* | Specifies the real user ID of the user for whom the table is being loaded and who will be authenticated to run the user space job. |
| *pid* | Specifies the process ID (pid) of the calling process, typically a job scheduler. |
| *job_key* | Specifies a globally unique job key. This job key is used later by the user space job to access the table. It must be greater than 0 and less than 0xFFF0. |
| *job_desc* | Specifies the string that describes the job that is using the network table. It has a maximum length of 50 characters that is truncated after the maximum. This string may contain blanks. It defaults to **no_job_description_given**. |

*window_memory* The input specifies the requested device memory to be allocated for each window in the network table. The value is represented in bytes.

This parameter returns the actual memory value allocated to each of the windows in the network table.

*table_size* Specifies the number of entries in the network table.

*table* Specifies the pointer to the network table structure array that was allocated and defined by the caller. The number of network table structures defined must be equal to *table_size*. The network table structure members are the following:

**unsigned short task_id**
Specifies a non-negative identifier that represents this task.

**unsigned short window_id**
Specifies the window ID to be loaded and used by the corresponding task ID.

**unsigned short lid**
Specifies the logical identifier of the SNI device upon which the window resides.

## Return Values

Upon successful completion, the **ntbl_load_table** subroutine returns a value of **NTBL_SUCCESS**.

Otherwise, it returns an integer value defined by the **NTBL_RC** enumerator found in the **ntbl.h** header file.

| | |
|---|---|
| **NTBL_SUCCESS** | Table was successfully loaded. |
| **NTBL_EPERM** | Caller does not have root user authority. |
| **NTBL_EINVAL** | Invalid input parameter. |
| **NTBL_EADAPTER** | Open failed on the *adapter* parameter. |
| **NTBL_EIOCTL** | The ioctl call failed. |
| **NTBL_ELID** | No lid was specified in the **NTBL** structure for the SNI device. |
| **NTBL_EMEM** | Failed to allocate memory. |
| **NTBL_EIO** | SNI device reported down state. |
| **NTBL_BUSY_STATE** | Window within the table was in use. |
| **NTBL_LOADED_STATE** | Window is already loaded. |
| **NTBL_DISABLED_STATE** | Window is disabled. |
| **NTBL_ACTIVE_STATE** | Window is active. |

# ntbl_load_table_rdma Subroutine

## Purpose

Loads a network table that has optional access to RDMA resources on the node from which it is invoked.

## Library

Network Table Library (**libntbl.a**)

## Syntax

```
#include <ntbl.h>

int ntbl_load_table_rdma (version, adapter, network_id, uid, pid, job_key, job_desc, window_memory, use_rdma, rcxt_blks, table_size, table)
int version;
char *adapter;
unsigned short network_id;
uid_t uid;
pid_t pid;
unsigned short job_key;
char *job_desc;
unsigned long long *window_memory;
```

```
int use_rdma;
int rcxt_blks;
int table_size;
NTBL **table;
```

## Description

The **ntbl_load_table_rdma** subroutine loads the network table on the node from which it is invoked. This network table is used by parallel jobs running user space over the switch.

An **NTBL** structure must be defined for every task in the parallel job. Each structure defines the task, window, and SNI device relationship. The array of structures defines how the tasks of the parallel job will communicate. The windows specified must be valid and in the **NTBL_UNLOADED_STATE** state.

Valid *window_memory* ranges can be obtained from the **ntbl_adapter_resources** subroutine. The *window_memory* requested is not guaranteed to be satisfied by the device driver, thus this parameter will record the allocated memory upon return. A *window_memory* input value of 0 indicates that the device driver will determine the window memory size. This value will be the value of the **win_min_size** member set by the administrator using the **chgsni** command.

You must have root user authority to invoke this subroutine.

## Parameters

| | |
|---|---|
| *version* | Specifies the version of header file used. This value should be **NTBL_VERSION**, which is defined in the **ntbl.h** header file. |
| *adapter* | Specifies the SNI device name to be opened. |
| *network_id* | Specifies the network ID for the SNI device. |
| *uid* | Specifies the real user ID of the user for whom the table is being loaded and who will be authenticated to run the user space job. |
| *pid* | Specifies the process ID (pid) of the calling process, typically a job scheduler. |
| *job_key* | Specifies a globally unique job key. This job key is used later by the user space job to access the table. It must be greater than 0 and less than 0xFFF0. |
| *job_desc* | Specifies the string that describes the job that is using the network table. It has a maximum length of 50 characters that is truncated after the maximum. This string may contain blanks. It defaults to **no_job_description_given**. |
| *window_memory* | The input specifies the requested device memory to be allocated for each window in the network table. The value is represented in bytes.<br><br>This parameter returns the actual memory value allocated to each of the windows in the network table. |
| *table_size* | Specifies the number of entries in the network table. |
| *use_rdma* | Indicates that the communication protocols using windows are allowed to use a Remote Direct Memory Access (RDMA) protocol when transferring data from one task to another. When this flag is set to 1, a single block of rCxt is made available to the protocol. The only other valid value is 0, which indicates that the RDMA protocol can not be used. Only four user space jobs can use the RDMA protocol at a given time in a single operating system image. |

*rcxt_blks*

The adapter resource rCxt is used to manage an RDMA operation. Each concurrent RDMA operation requires an rCxt resource at the initiator and target side of an RDMA transfer. An rCxt block consists of 128 units. The number of rCxt blocks available on an adapter is limited.

The function of the *rcxt_blks* parameter is dependent on the following settings of the *use_rdma* parameter:

- If the *use_rdma* parameter is set to 0 and the *rcxt_blks* parameter is set to 0, then the communication protocol using a window has no access to RDMA resources.
- If the *use_rdma* parameter is set to 1 and the *rcxt_blks* parameter is set to 0, then the communication protocol using a window can use RDMA and has access to one block of rCxt.
- If the *use_rdma* parameter is set to 0 and the *rcxt_blks* parameter is set to *n* (where $n > 0$), then the communication protocol using a window has no access to RDMA resources.
- If the *use_rdma* parameter is set to 1 and the *rcxt_blks* parameter is set to *n* (where $n > 0$), then the communication protocol using a window can use RDMA and has access to $n + 1$ rCxt blocks. If *n* is larger than the number of available blocks of rCxt on the adapter, the load will fail with a return code of **NTBL_NO_RDMA_AVAIL**.

*table*

Specifies the pointer to the network table structure array that was allocated and defined by the caller. The number of network table structures defined must be equal to *table_size*. The network table structure members are the following:

**unsigned short task_id**
Specifies a non-negative identifier that represents this task.

**unsigned short window_id**
Specifies the window ID to be loaded and used by the corresponding task ID.

**unsigned short lid**
Specifies the logical identifier of the SNI device upon which the window resides.

## Return Values

Upon successful completion, the **ntbl_load_table_rdma** subroutine returns a value of **NTBL_SUCCESS**.

Otherwise, it returns an integer value defined by the **NTBL_RC** enumerator found in the **ntbl.h** header file.

| | |
|---|---|
| **NTBL_SUCCESS** | Table was successfully loaded. |
| **NTBL_EPERM** | Caller does not have root user authority. |
| **NTBL_EINVAL** | Invalid input parameter. |
| **NTBL_EADAPTER** | Open failed on the *adapter* parameter. |
| **NTBL_EIOCTL** | The ioctl call failed. |
| **NTBL_ELID** | No lid was specified in the **NTBL** structure for the SNI device. |
| **NTBL_EMEM** | Failed to allocate memory. |
| **NTBL_EIO** | SNI device reported down state. |
| **NTBL_BUSY_STATE** | Window within the table was in use. |
| **NTBL_LOADED_STATE** | Window is already loaded. |
| **NTBL_DISABLED_STATE** | Window is disabled. |
| **NTBL_ACTIVE_STATE** | Window is active. |
| **NTBL_NO_RDMA_AVAIL** | This value can indicate any of the following:<br>• The device driver was not enabled for RDMA.<br>• More than four separate **NTBL** loads with different job IDs attempted to set *use_rdma*.<br>• More rCxt blocks were requested than are available. |

# ntbl_query_window Subroutine

## Purpose

Returns the current state of the network table window.

## Library

Network Table Library (**libntbl.a**)

## Syntax

```
#include <ntbl.h>

int ntbl_query_window (version, adapter, window_id, status)
int version;
char *adapter;
unsigned short window_id;
int *status;
```

## Description

The **ntbl_query_window** subroutine queries the current state of a network table window on the node from which it is invoked.

## Parameters

*version* Specifies the version of header file used. This value should be **NTBL_VERSION**, which is defined in the **ntbl.h** header file.

*adapter* Specifies the SNI device device name to be opened.

*window_id* Specifies the window ID for which the query will be done.

*status* The output contains the current state of the window specified. The state will be one of the following:

**NTBL_UNLOADED_STATE**
Window is not loaded and available.

**NTBL_LOADED_STATE**
Window is loaded but not opened yet.

**NTBL_DISABLED_STATE**
Window has been disabled.

**NTBL_BUSY_STATE**
Window is in a transitional state and cannot be enabled or disabled until it reaches a defined state.

**NTBL_ACTIVE_STATE**
Window is fully active and can process window disable requests.

## Return Values

Upon successful completion, the **ntbl_query_window** subroutine returns a value of **NTBL_SUCCESS**.

Otherwise, it returns an error value defined by the **NTBL_RC** enumerator found in the **ntbl.h** header file.

**NTBL_SUCCESS** Query was successful.
**NTBL_EINVAL** Invalid input parameter.
**NTBL_EADAPTER** Open failed on the *adapter* parameter.
**NTBL_EIOCTL** The ioctl call failed.
**NTBL_EIO** SNI device reported down state

# ntbl_status_adapter Subroutine

## Purpose

Returns the status of all network table windows on a specified SNI device.

## Library

Network Table Library (**libntbl.a**)

## Syntax

```
#include <ntbl.h>

int ntbl_status_adapter (version, adapter, window_count, status_info)
int version;
char *adapter;
int *window_count;
NTBL_STATUS **status_info;
```

## Description

The **ntbl_status_adapter** subroutine returns the status of all network table windows on a specified SNI device on the node from which it is invoked. It will allocate all **NTBL_STATUS** structures depending on the number of windows defined for the SNI device. These structures will be linked to the first **NTBL_STATUS** pointer by the next pointer. The caller is responsible for releasing the **NTBL_STATUS**-linked list using the **free** subroutine.

A null next pointer indicates the end of the list. The *window_count* parameter indicates the number of **NTBL_STATUS** structures in the linked list. If a window is loaded, the corresponding **NTBL_STATUS** structure will contain information about who made the load request, which user was designated to use the table, and when the request was made. If a window is not loaded or an error occurred, the *return_code* structure member contains the corresponding error value.

## Parameters

| | |
|---|---|
| *version* | Specifies the version of header file used. This value should be **NTBL_VERSION**, which is defined in the **ntbl.h** header file. |
| *adapter* | Specifies the SNI device device name for which status will be returned. |
| *window_count* | The output of the *window_count* parameter specifies the number of entries in the *status_info* linked list. |
| *status_info* | Specifies the address of a **NTBL_STATUS** pointer. This pointer will become the first entry in the linked list. |

## Return Values

Upon successful completion, the **ntbl_status_adapter** subroutine returns a value of **NTBL_SUCCESS**.

The following **NTBL_STATUS** data members are returned when the network table window is loaded, disabled, active, or busy:

| | |
|---|---|
| *user_name* | Name corresponding to the user ID given during the load request. |
| *description* | String given during the load request describing the job that is using the network table. |
| *time_loaded* | Timestamp of when the load request was processed. |
| *client_pid* | Process ID of the process that made the load request. |
| *uid* | User ID given during the load request for the user who will be using the network table. |
| *adapter* | SNI device that the window resides on. |
| *window_id* | Window ID for which the data is being reported. |

| | |
|---|---|
| *memory_requested* | Window memory requested during the **ntbl_load_table** call for use by the task. |
| *memory_allocated* | Window memory allocated during the **ntbl_load_table** call for use by the task. |
| *state* | Current state of the window. |

The following **NTBL_STATUS** data members are returned when the network table window is unloaded:

| | |
|---|---|
| *adapter* | SNI device that the window resides on. |
| *window_id* | Window for which the data is being reported. |
| *state* | Current state of the window. |

Otherwise, the **ntbl_status_adapter** subroutine returns an integer value defined by the **NTBL_RC** enumerator found in the **ntbl.h** header file.

| | |
|---|---|
| **NTBL_SUCCESS** | Query was successful. |
| **NTBL_EINVAL** | Invalid input parameter. |
| **NTBL_EADAPTER** | Open failed on the *adapter* parameter. |
| **NTBL_EIOCTL** | The ioctl call failed. |
| **NTBL_EMEM** | Failed to allocate memory. |
| **NTBL_EIO** | SNI device reported down state. |

---

# ntbl_unload_window Subroutine

## Purpose

Unloads the network table window on the node from which it is invoked.

## Library

Network Table Library (**libntbl.a**)

## Syntax

```
#include <ntbl.h>

int ntbl_unload_window (version, adapter, job_key, windown_id)
int version;
char *adapter;
unsigned short job_key;
unsigned short window_id;
```

## Description

The **ntbl_unload_window** subroutine unloads the network table window on the node from which it is invoked. This network table is used by parallel jobs running user space over the switch. The **ntbl_unload_window** subroutine checks that the *job_key* parameter value matches the job key value that was stored during the **ntbl_load_table** subroutine call.

If the *job_key* parameter does not match, the **ntbl_unload_window** subroutine returns an **NTBL_EPERM** value. Each window within the network table must be unloaded individually. The **ntbl_unload_window** subroutine should be called even when the SNI device reports **NTBL_EIO**. The resources held by a window will be released regardless of the state of the SNI device.

You must have root user authority to invoke this subroutine.

## Parameters

| | |
|---|---|
| *version* | Specifies the version of header file used. This value is the **NTBL_VERSION** version, which is defined in the **ntbl.h** header file. |
| *adapter* | Specifies the SNI device device name to be opened. |
| *job_key* | Specifies the globally unique job key that was provided when the **ntbl_load_table** subroutine was invoked. The *job_key* parameter must match this value in order to unload the window, and must be greater than 0 and less than 0xFFF0. |
| *window_id* | Specifies the window ID to be unloaded. |

## Return Values

Upon successful completion, the **ntbl_unload_window** subroutine returns a value of **NTBL_SUCCESS**.

Otherwise, it returns an integer value defined by the **NTBL_RC** enumerator found in the **ntbl.h** header file.

| | |
|---|---|
| **NTBL_SUCCESS** | Window was successfully unloaded. |
| **NTBL_EPERM** | The caller did not have root user authority. The *job_key* parameter did not match. |
| **NTBL_EINVAL** | Invalid input parameter. |
| **NTBL_EADAPTER** | Open failed on the *adapter* parameter. |
| **NTBL_ESYSTEM** | System error. |
| **NTBL_EIOCTL** | The **ioctl** call failed. |
| **NTBL_BUSY_STATE** | Window is in use. |
| **NTBL_UNLOADED_STATE** | Window is not loaded. |
| **NTBL_ACTIVE_STATE** | Window is active. |

# ntbl_version Subroutine

## Purpose

Returns the **NTBL_VERSION** value.

## Library

Network Table Library (**libntbl.a**)

## Syntax

```
#include <ntbl.h>

int ntbl_version (void);
```

## Description

The **ntbl_version** subroutine gets the value of the **NTBL_VERSION** value from the **libntbl.a** library. This represents the version of the interfaces for the **libntbl.a** library. This subroutine can be used to check the version of the library and determine if any interfaces have changed between versions.

## Return Values

The **ntbl_version** subroutine returns the **NTBL_VERSION** value.

# Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Dept. LRAS/Bldg. 003
11400 Burnet Road
Austin, TX 78758-3498
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

## Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX
AIX 5L
@server
IBM
LoadLeveler
pSeries
Redbooks 

Other company, product, or service names may be the trademarks or service marks of others.

# Index


## A
adapter_def resource 9
adapter_duplex attribute 9
adapter_speed attribute 9
attributes attribute 9

## C
cable_type attribute 8
chgmlt command 15
chgsni command 16
chgsni error messages 13
commands
  chgmlt 15
  chgsni 16
  defmlt 18
  getadapters 7
  nimadapters 7
  ntblclean 19
  ntblstatus 21
comments attribute 9
configuring
  multi-link interface 10
  NIM server 6
  prerequisites 5
  SNI 10

## D
defmlt command 18

## E
error messages
  chgsni 13

## F
filesets 4

## G
getadapters command 7

## H
hardware requirements 5

## I
install_gateway attribute 9
install_server attribute 9
interface_name attribute 9
ISO 9000 v

## J
job schedulers 3

## L
large pages 2
location attribute 9

## M
MAC_address attribute 9
machine_type attribute 8
media_speed attribute 9

## N
netaddr attribute 8
Network Installation Management (NIM) 6
network table services 3
network_type attribute 8
NIM 6
nimadapters 9
nimadapters command 7
ntbl_adapter_resources subroutine 23
ntbl_clean_window subroutine 24
ntbl_disable_window subroutine 26
ntbl_enable_window subroutine 27
ntbl_load_table subroutine 28
ntbl_load_table_rdma subroutine 29
ntbl_query_window subroutine 32
ntbl_status_adapter subroutine 33
ntbl_unload_window subroutine 34
ntbl_version subroutine 35
ntblclean command 19
ntblstatus command 21

## O
overview
  multi-link support 1
  network table services 3
  Remote Direct Memory Access 3
  switch network interface 1
  Technical Large Pages 2

## P
ping_status attribute 9
prerequisites 5

## R
RDMA 3
related information v
Remote Direct Memory Access 3

# S

secondary_hostname attribute 9
SNI
  support on AIX 1
software requirements 6
stanza file
  attributes 8
  create 7
  format 7
  gathering attributes for 7
subnet 10
subnet_mask attribute 8
subroutines
  ntbl_adapter_resources 23
  ntbl_clean_window 24
  ntbl_disable_window 26
  ntbl_enable_window 27
  ntbl_load_table 28
  ntbl_load_table_rdma 29
  ntbl_query_window 32
  ntbl_status_adapter 33
  ntbl_unload_window 34
  ntbl_version 35

# T

Technical Large Pages 2
text highlighting v
tuning 11

# U

using
  Remote Direct Memory Access 3

# Readers' Comments — We'd Like to Hear from You

**Switch Network Interface for eServer pSeries High Performance Switch Guide and Reference**

**Publication No. SC23-4869-04**

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send your comments via e-mail to: pserinfo@us.ibm.com

If you would like a response from IBM, please fill in the following information:

| | |
|---|---|
| Name | Address |
| Company or Organization | |
| Phone No. | E-mail address |

**Readers' Comments — We'd Like to Hear from You**
SC23-4869-04

IBM

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department 04XA-905-6B013
11501 Burnet Road
Austin, TX 78758-3400

Fold and Tape | **Please do not staple** | Fold and Tape



Cut or Fold
Along Line

Printed in U.S.A.

SC23-4869-04