

# **SANbox2 Simple Network Management Protocol Reference Manual**

Publication No. 59044-00 Rev. A

QLogic Corporation  
6321 Bury Drive  
Eden Prairie, MN 55346  
(952) 932-4000

## Revision Record

Date	Description
June 2002	Initial Release

Information furnished in this manual is believed to be accurate and reliable. However, QLogic Corporation assumes no responsibility for its use, nor for any infringements of patents or other rights of third parties which may result from its use. QLogic Corporation reserves the right to change product specifications at any time without notice. Applications described in this document for any of these products are for illustrative purposes only. QLogic Corporation makes no representation nor warranty that such applications are suitable for the specified use without further testing or modification. QLogic Corporation assumes no responsibility for any errors that may appear in this document.

Copyright © QLogic Corporation 2002. All rights reserved.

IBM is a trademark of International Business Machines.

Intel is a trademark of Intel Corporation.

Linux is a registered trademark of Linus Torvalds.

Microsoft, Windows NT, and Windows 2000, and Internet Explorer are trademarks of Microsoft Corporation.

Novell and NetWare are registered trademarks of Novell, Inc.

QLogic, SANbox, SANbox2, SANblade, and Multistage are trademarks or registered trademarks of QLogic Corporation.

Red Hat and all Red Hat-based trademarks are trademarks or registered trademarks of Red Hat, Inc.

Solaris is a registered trademark of Sun Microsystems, Inc.

SPARC is a registered trademark of SPARC International, Inc. Products bearing SPARC trademarks are based on an architecture developed by Sun Microsystems, Inc.

Symmetrix is a registered trademark of EMC Corporation.

Netscape Navigator is a registered trademark of Netscape Communications Corporation.

Brocade is a registered trademark of Brocade Communications Systems Inc.

All other brand and product names are trademarks or registered trademarks of their respective owners.

# Table of Contents

---

## **Preface**

Intended Audience	1
Related Materials	1
Technical Support	2
Availability	2
Training	2
Contact Information	2

---

## **1 Introduction**

SNMP Interface Objectives	1-1
Manager and Agent	1-1
Traps	1-3
Management Information Base	1-4
User Datagram Protocol	1-4
Numbering System Conventions	1-4

---

## **2 Configuring a Switch**

System Specifications and Requirements	2-1
Configuring a Switch Using Telnet	2-2
Configuring a Switch Using SANbox Manager	2-3

---

## **3 MIB-II Objects**

Groups in MIB-II	3-1
The System Group	3-1
The Interfaces Group	3-4
The Interfaces Table	3-5
The Address Translation Group	3-12
The IP Group	3-13
The IP Address Table	3-19
The IP Routing Table	3-21
The IP Address Translation Table	3-25
Additional IP Objects	3-27
The ICMP Group	3-27
The TCP Group	3-35
The TCP Connection Table	3-38
Additional TCP Objects	3-40
The UDP Group	3-41

---

The UDP Listener Table 3-42  
The EGP Group 3-43  
The EGP Neighbor Table 3-44  
The Transmission Group 3-49  
The SNMP Group 3-49

---

## **4 Fibre Alliance MIB Objects**

FA MIB Definitions 4-1  
Connectivity Unit Group 4-3  
Connectivity Table 4-5  
Revision Table 4-18  
Sensor Table 4-20  
Port Table 4-26  
Event Table 4-40  
Link Table 4-44  
Port Statistics Table 4-50  
Simple Name Server Table 4-69  
Trap Table 4-76  
Related Traps 4-79

---

## **5 Fabric Element MIB Objects**

Fibre Channel FE MIB Definitions 5-1  
Configuration Group 5-2  
Module Table 5-3  
FxPort Configuration Table 5-6  
The Status Group 5-11  
FxPort Physical Level Table 5-13  
Fx Port Fabric Login Table 5-16  
The Error Group 5-21  
Accounting Groups 5-25  
Class 2 Accounting Table 5-29  
Class 3 Accounting Table 5-32  
Capability Group 5-34

---

## **Glossary**

---

## **Index**

# List of Figures

---

Figure 1-1	SNMP Interface Architecture 1-2
Figure 4-1	connUnitEventDescr Variable Format 4-80

# List of Tables

---

Table 1-1	Trap Severity Levels 1-3
Table 3-1	MIB-II Groups 3-1
Table 4-1	FA-MIB Textual Substitutions 4-1
Table 4-2	Switch Operational States 4-7
Table 4-3	Connectivity Unit Return Values 4-8
Table 4-4	connUnitControl Read Return Values 4-15
Table 4-5	connUnitControl Write Control Values 4-15
Table 4-6	connUnitEventFilter Read Return Values 4-16
Table 4-7	connUnitEventFilter Control Write Values 4-17
Table 4-8	ConnUnitRevsRevId Return Values 4-19
Table 4-9	ConnUnitRevsDescription Return Values 4-20
Table 4-10	ConnUnitSensorName Return Values 4-21
Table 4-11	ConnUnitSensorStatus Return Values for Board Temperature 4-22
Table 4-12	ConnUnitSensorStatus Return Values for Fan Status 4-23
Table 4-13	ConnUnitSensorStatus Return Values for Power Supply Status 4-23
Table 4-14	ConnUnitSensorMessage Values 4-24
Table 4-15	ConnUnitSensorType Return Values 4-25
Table 4-16	ConnUnitSensorCharacteristic Values 4-25
Table 4-17	ConnUnitPortType Return Values 4-28
Table 4-18	ConnUnitPortState Return Values 4-30
Table 4-19	ConnUnitPortTransmitterType Return Values 4-31
Table 4-20	ConnUnitPortControl Read Return Values 4-36
Table 4-21	ConnUnitPortControl Write Command Values 4-36
Table 4-22	ConnUnitPortHWState Port State Return Values 4-40
Table 4-23	ConnUnitPortType State Return Values 4-73
Table 4-24	Trap Severity Levels 4-76
Table 4-25	connUnitEventDescr Variable Field Descriptions 4-80
Table 4-26	connUnitEventDescr Trap List 4-81
Table 5-1	FA-MIB Textual Substitutions 5-1
Table 5-2	Port Operational Modes 5-13
Table 5-3	Port Administrative Modes 5-13
Table 5-4	fcFxPortPhysAdminStatus Read Return Values 5-14
Table 5-5	fcFxPortPhysAdminStatus Write Values 5-14
Table 5-6	fcFxPortPHysOperStatus Return Values 5-15

# Preface

---

This manual describes the support for Simple Network Management Protocol (SNMP) used with the SANbox2 switch products. This Simple Network Management Protocol (SNMP) Reference Manual describes how to use SNMP to manage and monitor the SANbox2 switch products.

This manual is organized as follows:

- [Section 1](#) describes SNMP objectives, traps, Management Information Bases (MIB), and User Datagram Protocol.
- [Section 2](#) describes how to configure a SANbox2 switch using Telnet and the SANbox Manager graphical user interface.
- [Section 3](#) describes the Management Information Database (MIB-II).
- [Section 4](#) describes the Fabric Alliance - Management Information Database (FA-MIB).
- [Section 5](#) describes the Fibre Element - Management Information Database (FE-MIB).

## Intended Audience

This manual is intended for users responsible for the support of SNMP and SANbox2 switch configurations.

## Related Materials

Refer to the following manual for switch hardware and installation information:

- *SANbox2-16 Fibre Channel Switch Installer's/User's Manual*, publication number 59021-03.
- *SANbox2 Switch Management User's Manual*, publication number 59022-03.

## Technical Support

Customers should contact their authorized maintenance provider for technical support of their QLogic switch products. QLogic-direct customers may contact QLogic Technical Support; others will be redirected to their authorized maintenance provider.

Visit the QLogic switch support Web site listed in [“Contact Information”](#) for the latest firmware and software updates.

## Availability

QLogic Technical Support is available from 7:00 AM to 7:00 PM Central Standard Time, Monday through Friday, excluding QLogic-observed holidays.

## Training

QLogic offers the following technical training courses:

- Switch Certification
- HBA Certification

Each course is available at the training facility in Eden Prairie, MN or at your local facility. All courses include a Fibre Channel overview and sections on installation, maintenance, and topology solutions. Each student receives a set of manuals and a CD-ROM containing course training materials. Upon successful completion of the training, Qlogic awards a certificate identifying the student as a Certified SANbox™ or SANblade™ Professional.

## Contact Information

Address:	QLogic Switch Products Inc. 6321 Bury Drive Eden Prairie, Minnesota 55346 USA
Telephone:	+1 952-932-4040
Fax:	+1 952-932-4018
Email:	
Technical Service	support@qlogic.com
Technical Training	tech.training@qlogic.com
Switch Support Web Site:	www.qlogic.com/support/home_support.asp



Simple Network Management Protocol (SNMP) is the protocol governing network management and monitoring of network devices. This Simple Network Management Protocol Reference Manual describes how to use SNMP to manage and monitor the SANbox2 switch products. Specifically, this manual describes the SNMP agent that resides on the switch.

The following topics are covered in this section:

- SNMP interface objectives
- Manager and agent
- Traps
- Management information bases (MIBs)
- User datagram protocol (UDP)
- Numbering system conventions

## SNMP Interface Objectives

The objectives of the SNMP Interface are as follows:

- Connect to the SNMP agent that resides on the switch using a management workstation.
- Support of Fabric Element Management Information Bases (FE-MIB) (rfc2837) and Fibre Alliance Management Information Bases (FA-MIB) draft.
- Support of version 2 traps.
- The SNMP agent supports SNMPv1, SNMPv2c, and SNMPv3.

## Manager and Agent

The two primary elements of SNMP are:

- Manager - the application that runs on the management workstation.
- Agent - the daemon application that runs on the switch.

The Manager is the application through which the network administrator performs network management functions. The SNMP agent is the direct interface on the switch for any SNMP manager connecting to the switch using the SNMP protocol, as shown in [Figure 1-1](#). The agent will be started by the script file(s) responsible for switch initialization when the switch powers up or when the switch is reset.

When an SNMP request arrives at the agent, the agent will compose a message and pass it on to Switch Management to process the message and provide a response to the agent. The agent then provides a response to the originator of the SNMP request. The SNMP agent does not have direct access to the internal database of the switch.

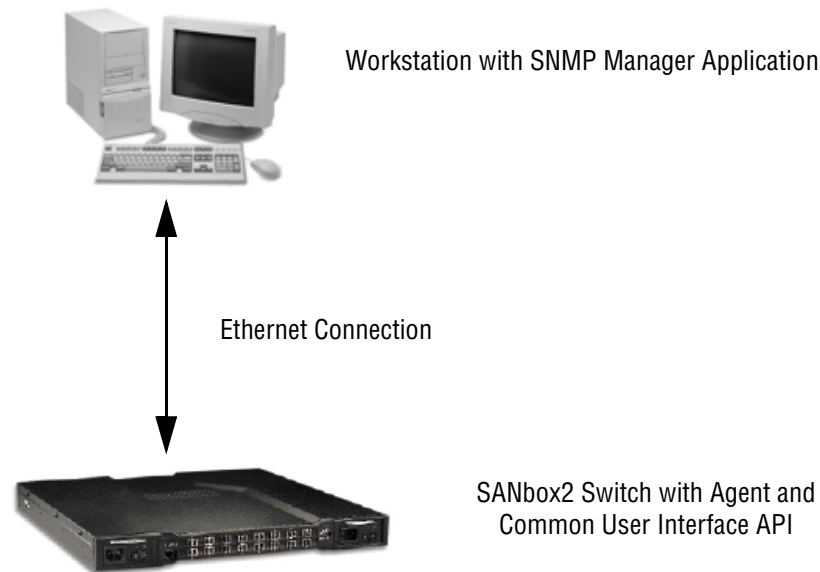


Figure 1-1 SNMP Interface Architecture

## Traps

Traps are notification messages sent from the switch to a registered manager when a change of state occurs within the switch. A change of state can be an alarm condition or simply a configuration change.

The Fibre Alliance MIB defines a trap table configurable through SNMP. A trap table may have up to 5 entries, and can be configured using the SNMP Manager or SANbox Manager graphical user interface. The same trap table information is available to both SNMP Manager and SANbox Manager.

A trap event is reported when the incoming error has a severity level less than or equal to the configured severity level. The trap event types and trap severity levels are listed in [Table 1-1](#). Refer to [“Trap Table” on page 4-76](#) for information on specific traps.

**Table 1-1 Trap Severity Levels**

Event Type	Severity Level
Unknown	1
Emergency	2
Alert	3
Critical	4
Error	5
Warning	6
Notify	7
Info	8
Debug	9
Mark	10

## Management Information Base

Management information bases (MIBs) are a collection of definitions that define the properties of the managed object within the device to be managed. Every managed device keeps a database of values for each definition written in the MIB. It is not the actual database itself, it is implementation dependant. Definition of the MIB conforms to the SMI given in Request For Comment (RFC) 1155. The latest Internet MIB is given in RFC 1213, and is sometimes called the MIB-II.

## User Datagram Protocol

SANbox2 supports the following User Datagram Protocol (UDP) settings:

- Agents “listen” on UDP port 161.
- Responses are sent back to the originating Network Management Station (NMS) port from a dynamic port, although many agents use port 161 also for this target.
- The maximum SNMP message size is 65507 octets (maximum UDP message size).
- The minimum receive packets size for SNMP implementations is 484 octets in length.
- Agent and Network Monitoring Systems are responsible for determining error recovery.

## Numbering System Conventions

The conventions for numbering systems in this manual are.

- Decimal= 101
- Hexadecimal= 0x101
- Binary= 101b

This section describes how to configure a SANbox2 switch to support SNMP. The following topics are covered:

- System specifications and requirements
- Configuring a switch using the Telnet command line interface
- Configuring a switch using the SANbox Manager software application

## System Specifications and Requirements

- SANbox2 switches support SNMP versions 1, 2, and 3.
- Version 2 traps are supported.
- Hardware - one out-of-band Ethernet connection is required.
- Software - one switch management software application allows you to:
  - Monitor and control the switch.
  - Read, write, and receive trap information, if supported.
- Ports on the switch reserved for SNMP:
  - Port 161 is not configurable, and is used for the standard SNMP commands.
  - Port 162 is configurable and is the default port used for traps.
- One or more in-band switches can be managed by an out-of-band SANbox2 switch acting as a proxy switch.
- SANbox2 can only act as a proxy for other SANbox2 switches.

## Configuring a Switch Using Telnet

The Telnet command line interface offers a convenient way to change SNMP parameters. SNMP parameter defaults are preset during manufacturing. For security purposes, these default values should be changed.

For specific information about SNMP parameters, refer to the SNMP Configuration section in the *SANbox2 Switch Management User's Manual*.

To configure a SANbox2 switch using the Telnet, do the following. Press the Enter key to accept the default value for each parameter.

```
cli #> admin start
cli (admin) #> set setup snmp
```

A list of attributes with formatting and current values will follow.

Enter a new value or simply press the ENTER key to accept the default value.

If you wish to terminate this process before reaching the end of the list press 'q' or 'Q' and the ENTER key to do so.

Trap Severity Options

```
-----
unknown, emergency, alert, critical, error, warning, notify, info, debug
Contact          (string, max=32 chars)      [Customer      ]
Location         (string, max=32 chars)      [TheLab        ]
Trap1Enabled     (True / False)             [True          ]
Trap1Address     (dot-notated IP Address)   [10.20.71.15   ]
Trap1Port        (decimal value)           [162           ]
Trap1Severity    (see allowed options above) [warning       ]
Trap2Enabled     (True / False)             [False         ]
Trap2Address     (dot-notated IP Address)   [0.0.0.0       ]
Trap2Port        (decimal value)           [162           ]
Trap2Severity    (see allowed options above) [warning       ]
Trap3Enabled     (True / False)             [False         ]
Trap3Address     (dot-notated IP Address)   [0.0.0.0       ]
Trap3Port        (decimal value)           [162           ]
Trap3Severity    (see allowed options above) [warning       ]
Trap4Enabled     (True / False)             [False         ]
Trap4Address     (dot-notated IP Address)   [0.0.0.0       ]
Trap4Port        (decimal value)           [162           ]
Trap4Severity    (see allowed options above) [warning       ]
Trap5Enabled     (True / False)             [False         ]
Trap5Address     (dot-notated IP Address)   [0.0.0.0       ]
Trap5Port        (decimal value)           [162           ]
Trap5Severity    (see allowed options above) [warning       ]
ReadCommunity    (string, max=32 chars)     [public        ]
WriteCommunity   (string, max=32 chars)     [private       ]
```

```
TrapCommunity      (string, max=32 chars)      [public      ]
AuthFailureTrap    (True / False)              [False       ]
```

```
Do you want to save and activate this snmp setup? (y/n) [n] y
```

```
SNMP setup saved and activated.
```

## Configuring a Switch Using SANbox Manager

To configure a SANbox2 switch using SANbox Manager, use the Switch Properties and Network Properties windows. For specific information, refer to the *SANbox2 Switch Management User's Manual*, publication number 59022-03.

## Notes



This section covers the implementation details for the MIB-II on the SANbox2 switch. A MIB (management information base) is a collection of definitions that define the properties of the managed object within the device to be managed. Every managed device keeps a database of values for each definition written in the MIB. It is not the actual database itself, it is implementation dependant. Definition of the MIB conforms to the SMI given in RFC 1155. The latest Internet MIB is given in RFC 1213, and is sometimes called the MIB-II.

### Groups in MIB-II

Refer the [Table 3-1](#) for the syntax for MIB-II Groups.

**Table 3-1 MIB-II Groups**

Group	Syntax
system	OBJECT IDENTIFIER ::= { mib-2 1 }
interfaces	OBJECT IDENTIFIER ::= { mib-2 2 }
at	OBJECT IDENTIFIER ::= { mib-2 3 }
ip	OBJECT IDENTIFIER ::= { mib-2 4 }
icmp	OBJECT IDENTIFIER ::= { mib-2 5 }
tcp	OBJECT IDENTIFIER ::= { mib-2 6 }
udp	OBJECT IDENTIFIER ::= { mib-2 7 }
historical	
cmot	OBJECT IDENTIFIER ::= { mib-2 9 }
transmission	OBJECT IDENTIFIER ::= { mib-2 10 }
snmp	OBJECT IDENTIFIER ::= { mib-2 11 }

### The System Group

Implementation of the System group is mandatory for all systems. If an agent is not configured to have a value for any of these variables, a string of length 0 is returned.

---

**sysDescr**

A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating-system, and networking software. It is mandatory that this only contain printable ASCII (American Standard Code for Information Interchange) characters.

**Syntax**

DisplayString (SIZE (0..255))

**Access**

read-only

**Status**

Mandatory

**Return Value**

The default is: QLogic SANbox2 FC Switch

**sysObjectID**

The vendor's authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining 'what kind of box' is being managed. For example, if vendor 'Flintstones, Inc.' was assigned the subtree 1.3.6.1.4.1.4242, it could assign the identifier 1.3.6.1.4.1.4242.1.1 to its 'Fred Router'.

**Syntax**

OBJECT IDENTIFIER

**Access**

read-only

**Status**

mandatory

**Return Value**

1.3.6.1.4.1.1663.1.1.1.11

**sysUpTime**

The time, in hundredths of a second, since the network management portion of the system was last reinitialized.

**Syntax**

TimeTicks

**Access**

read-only

**Status**

mandatory

**Return Value**

The time since last Power On Self Test (POST) was executed (for example, 3 days 21 hours).

**sysContact**

The textual identification of the contact person for this managed Node, together with information on how to contact this person.

**Syntax**

DisplayString (SIZE (0..255))

**Access**

read-write

**Status**

mandatory

**Return Value**

The default is: <sysContact undefined>

**sysName**

An administratively-assigned name for this managed Node. By convention, this is the Node's fully-qualified domain name.

**Syntax**

DisplayString (SIZE (0..255))

**Access**

read-write

**Status**

mandatory

**Return Value**

Default is: SANbox2

## sysLocation

The physical location of this Node, such as telephone closet and 3rd floor.

### Syntax

DisplayString (SIZE (0..255))

### Access

read-write

### Status

mandatory

### Return Value

The default is: <sysLocation undefined>

## sysServices

A value which indicates the set of services that this entity primarily offers. The value is a sum. This sum initially takes the value zero. Then, for each layer L in the range 1 through 7 that this Node performs transactions for, 2 raised to (L - 1) is added to the sum. For example, a Node which performs primarily routing functions would have a value of 4 ( $2^{(3-1)}$ ). In contrast, a Node which is a host offering application services would have a value of 72 ( $2^{(4-1)} + 2^{(7-1)}$ ).

### Syntax

INTEGER (0..127)

### Access

read-only

### Status

mandatory

### Return Value

The default is: 2

## The Interfaces Group

Implementation of the Interfaces group is mandatory for all systems.

## ifNumber

The number of network interfaces (regardless of their current state) present on this system.

### Syntax

INTEGER

**Access**

read-only

**Status**

mandatory

**Return Value**

The default is: 1

## The Interfaces Table

The Interfaces table contains information on the entity's interfaces. Each interface is thought of as being attached to a 'subnetwork'. This term should not be confused with 'subnet' which refers to an addressing partitioning scheme used in the Internet suite of protocols.

### ifIndex

A unique value for each interface. Its value ranges between 1 and the value of ifNumber. The value for each interface must remain constant at least from one re-initialization of the entity's network management system to the next reinitialization.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

### ifDescr

A textual string containing information about the interface. This string should include the name of the manufacturer, the product name, and the version of the hardware interface.

**Syntax**

DisplayString (SIZE (0..255))

**Access**

read-only

**Status**

mandatory

### ifType

The type of interface distinguished according to the physical/link protocol(s) immediately ‘below’ the network layer in the protocol stack.

#### Syntax

INTEGER

#### Access

read-only

#### Status

mandatory

### ifMtu

The size of the largest datagram which can be sent/received on the interface, specified in octets. For interfaces that are used for transmitting network datagrams, this is the size of the largest network datagram that can be sent on the interface.

#### Syntax

INTEGER

#### Access

read-only

#### Status

mandatory

### ifSpeed

An estimate of the interface’s current bandwidth in bits per second. For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal bandwidth.

#### Syntax

Gauge

#### Access

read-only

#### Status

mandatory

**ifPhysAddress**

The interface's address at the protocol layer immediately “below” the network layer in the protocol stack. For interfaces which do not have such an address, such as a serial line, this object should contain an octet string of zero length.

**Syntax**

PhysAddress

**Access**

read-only

**Status**

mandatory

**ifAdminStatus**

The desired state of the interface. The testing(3) state indicates that no operational packets can be passed.

**Syntax**

INTEGER

**Access**

read-write

**Status**

mandatory

**ifOperStatus**

The current operational state of the interface. The testing(3) state indicates that no operational packets can be passed.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**ifLastChange**

The value of sysUpTime at the time the interface entered its current operational state. If the current state was entered prior to the last reinitialization of the local network management subsystem, then this object contains a zero value.

## The Interfaces Table

---

### Syntax

TimeTicks

### Access

read-only

### Status

mandatory

## ifInOctets

The total number of octets received on the interface, including framing characters.

### Syntax

Counter

### Access

read-only

### Status

mandatory

## ifInUcastPkts

The number of subnetwork-unicast packets delivered to a higher-layer protocol.

### Syntax

Counter

### Access

read-only

### Status

mandatory

## ifInNUcastPkts

The number of non-unicast (that is, subnetwork- broadcast or subnetwork-multi-cast) packets delivered to a higher-layer protocol.

### Syntax

Counter

### Access

read-only



**Status**

mandatory

**ifInDiscards**

The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**ifInErrors**

The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**ifInUnknownProtos**

The number of packets received from the interface that were discarded because of an unknown or unsupported protocol.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

### ifOutOctets

The total number of octets transmitted out of the interface, including framing characters.

#### Syntax

Counter

#### Access

read-only

#### Status

mandatory

### ifOutUcastPkts

The total number of packets that higher level protocols requested be transmitted to a subnetwork unicast address, including those that were discarded or not sent.

#### Syntax

Counter

#### Access

read-only

#### Status

mandatory

### ifOutNUcastPkts

The total number of packets that higher level protocols requested be transmitted to a non-unicast (subnetwork broadcast or subnetwork multicast) address, including those that were discarded or not sent.

#### Syntax

Counter

#### Access

read-only

#### Status

mandatory

### ifOutDiscards

The number of outbound packets that were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**ifOutErrors**

The number of outbound packets that could not be transmitted because of errors.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**ifOutQLen**

The length of the output packet queue (in packets).

**Syntax**

Gauge

**Access**

read-only

**Status**

mandatory

**ifSpecific**

A reference to MIB definitions specific to the particular media being used to realize the interface. For example, if the interface is realized by an Ethernet, then the value of this object refers to a document defining objects specific to Ethernet. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntactically valid object identifier, and any conformant implementation of ASN.1 (Abstract Syntax Notation) and BER must be able to generate and recognize this value.

**Syntax**

OBJECT IDENTIFIER

### **Access**

read-only

### **Status**

mandatory

## **The Address Translation Group**

Implementation of the Address Translation group is mandatory for all systems. However, this group is deprecated by MIB-II. That is, it is being included solely for compatibility with MIB-I Nodes, and will most likely be excluded from MIB-III Nodes. From MIB-II and onwards, each network protocol group contains its own address translation tables.

The Address Translation group contains one table which is the union across all interfaces of the translation tables for converting a NetworkAddress (for example, an IP address) into a subnetwork-specific address. For lack of a better term, this document refers to such a subnetwork-specific address as a ‘physical’ address.

Examples of such translation tables are for broadcast media where ARP is in use, the translation table is equivalent to the ARP cache, or on an X.25 network where non-algorithmic translation to X.121 addresses is required. The translation table contains the NetworkAddress to X.121 address equivalences.

### **atIfIndex**

The interface on which this entry’s equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

### **Syntax**

INTEGER

### **Access**

read-write

### **Status**

deprecated

### **atPhysAddress**

The media-dependent “physical” address. Setting this object to a null string (one of zero length) has the effect of invalidating the corresponding entry in the atTable object. That is, it effectively disassociates the interface identified with the entry from the mapping identified with the entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management workstations must be prepared to receive tabular information from

agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant atPhysAddress object.

**Syntax**

PhysAddress

**Access**

read-write

**Status**

deprecated

**atNetAddress**

The NetworkAddress corresponding to the media-dependent ‘physical’ address.

**Syntax**

NetworkAddress

**Access**

read-write

**Status**

deprecated

**The IP Group**

Implementation of the IP group is mandatory for all systems.

**ipForwarding**

The indication of whether this entity is acting as an IP Gateway with respect to the forwarding of datagrams received by, but not addressed to, this entity. IP Gateways forward datagrams; IP hosts do not (except those source-routed from the host).

For some managed Nodes, this object may take on only a subset of the values possible. Accordingly, it is appropriate for an agent to return a “badValue” response if a management station attempts to change this object to an inappropriate value.

**Syntax**

INTEGER

**Access**

read-write

**Status**

mandatory

---

**ipDefaultTTL**

The default value inserted into the Time-To-Live field of the IP header of datagrams originated at this entity whenever a TTL value is not supplied by the transport layer protocol.

**Syntax**

INTEGER

**Access**

read-write

**Status**

mandatory

**ipInReceives**

The total number of input datagrams received from interfaces, including those received in error.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**ipInHdrErrors**

The number of input datagrams discarded due to errors in their IP headers. These include bad checksums, version number mismatch, other format errors, time-to-live exceeded, and errors discovered in processing their IP options.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**ipInAddrErrors**

The number of input datagrams discarded because the IP address in their IP header's destination field was not a valid address to be received at this entity. This

count includes invalid addresses (for example, 0.0.0.0) and addresses of unsupported Classes (for example, Class E). For entities which are not IP Gateways and therefore do not forward datagrams, this counter includes datagrams discarded because the destination address was not a local address.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**ipForwDatagrams**

The number of input datagrams for which this entity was not their final IP destination. As a result, an attempt was made to find a route to forward them to that final destination. In entities that do not act as IP Gateways, this counter will include only those packets that were Source Routed from this entity, and the Source Route option processing was successful.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**ipInUnknownProtos**

The number of locally-addressed datagrams received successfully but discarded because of an unknown or unsupported protocol.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

## ipInDiscards

The number of input IP datagrams for which no problems were encountered to prevent their continued processing, but which were discarded (for example, for lack of buffer space). This counter does not include any datagrams discarded while awaiting reassembly.

### Syntax

Counter

### Access

read-only

### Status

mandatory

## ipInDelivers

The total number of input datagrams successfully delivered to IP user protocols (including ICMP).

### Syntax

Counter

### Access

read-only

### Status

mandatory

## ipOutRequests

The total number of IP datagrams that local IP user protocols (including ICMP) supplied to IP in requests for transmission. This counter does not include any datagrams counted in ipForwDatagrams.

### Syntax

Counter

### Access

read-only

### Status

mandatory



## ipOutDiscards

The number of output IP datagrams for which no problem was encountered to prevent their transmission to their destination, but which were discarded (for example, for lack of buffer space). This counter would include datagrams counted in ipForwDatagrams if any such packets met this (discretionary) discard criterion.

### Syntax

Counter

### Access

read-only

### Status

mandatory

## ipOutNoRoutes

The number of IP datagrams discarded because no route could be found to transmit them to their destination. This counter includes any packets counted in ipForwDatagrams which meet this “no-route” criterion. This includes any datagrams that a host cannot route because all of its default gateways are down.

### Syntax

Counter

### Access

read-only

### Status

mandatory

## ipReasmTimeout

The maximum number of seconds which received fragments are held while they are awaiting reassembly at this entity.

### Syntax

INTEGER

### Access

read-only

### Status

mandatory

---

**ipReasmReqds**

The number of IP fragments received that needed to be reassembled at this entity.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**ipReasmOKs**

The number of IP datagrams successfully reassembled.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**ipReasmFails**

The number of failures detected by the IP reassembly algorithm for example, timed out, errors). This is not necessarily a count of discarded IP fragments, since some algorithms (notably the algorithm in RFC 815) can lose track of the number of fragments by combining them as they are received.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**ipFragOKs**

The number of IP datagrams that have been successfully fragmented at this entity.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**ipFragFails**

The number of IP datagrams that have been discarded because they needed to be fragmented at this entity, but could not because their Don't Fragment flag was set.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**ipFragCreates**

The number of IP datagram fragments that have been generated as a result of fragmentation at this entity.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**The IP Address Table**

The IP address table contains this entity's IP addressing information.

**ipAdEntAddr**

The IP address to which this entry's addressing information pertains.

**Syntax**

IpAddress

**Access**

read-only

**Status**

mandatory

**ipAdEntIfIndex**

The index value which uniquely identifies the interface to which this entry is applicable. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**ipAdEntNetMask**

The subnet mask associated with the IP address of this entry. The value of the mask is an IP address with all the network bits set to 1 and all the hosts bits set to 0.

**Syntax**

IpAddress

**Access**

read-only

**Status**

mandatory

**ipAdEntBcastAddr**

The value of the least-significant bit in the IP broadcast address used for sending datagrams on the (logical) interface associated with the IP address of this entry. For example, when the Internet standard all-ones broadcast address is used, the value will be 1. This value applies to both the subnet and network broadcasts addresses used by the entity on this (logical) interface.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**ipAdEntReasmMaxSize**

The size of the largest IP datagram which this entity can reassemble from incoming IP fragmented datagrams received on this interface.

**Syntax**

INTEGER (0..65535)

**Access**

read-only

**Status**

mandatory

**The IP Routing Table**

The IP routing table contains an entry for each route presently known to this entity.

**ipRouteDest**

The destination IP address of this route. An entry with a value of 0.0.0.0 is considered a default route. Multiple routes to a single destination can appear in the table, but access to such multiple entries is dependent on the table-access mechanisms defined by the network management protocol in use.

**Syntax**

IpAddress

**Access**

read-write

**Status**

mandatory

**ipRouteIfIndex**

The index value which uniquely identifies the local interface through which the next hop of this route should be reached. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

**Syntax**

INTEGER

**Access**

read-write

### Status

mandatory

## ipRouteMetric1

The primary routing metric for this route. The semantics of this metric are determined by the routing protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

### Syntax

INTEGER

### Access

read-write

### Status

mandatory

## ipRouteMetric2

An alternate routing metric for this route. The semantics of this metric are determined by the routing protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

### Syntax

INTEGER

### Access

read-write

### Status

mandatory

## ipRouteMetric3

An alternate routing metric for this route. The semantics of this metric are determined by the routing protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

### Syntax

INTEGER

### Access

read-write

### Status

mandatory

## ipRouteMetric4

An alternate routing metric for this route. The semantics of this metric are determined by the routing protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

### Syntax

INTEGER

### Access

read-write

### Status

mandatory

## ipRouteNextHop

The IP address of the next hop of this route. In the case of a route bound to an interface which is realized from a broadcast media, the value of this field is the agent's IP address on that interface.

### Syntax

IpAddress

### Access

read-write

### Status

mandatory

## ipRouteType

The type of route. The values direct(3) and indirect(4) refer to the notion of direct and indirect routing in the IP architecture. Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipRouteTable object. That is, it effectively disassociates the destination identified with the entry from the route identified with the entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipRouteType object.

### Syntax

INTEGER

### Access

read-write

**Status**

mandatory

**ipRouteProto**

The routing mechanism through which this route was learned. Inclusion of values for gateway routing protocols is not intended to imply that hosts should support those protocols.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**ipRouteAge**

The number of seconds since this route was last updated or otherwise determined to be correct. No semantics of 'too old' can be implied except through knowledge of the routing protocol by which the route was learned.

**Syntax**

INTEGER

**Access**

read-write

**Status**

mandatory

**ipRouteMask**

Indicate the mask to be logical-ANDed with the destination address before being compared to the value in the ipRouteDest field.

**Syntax**

IpAddress

**Access**

read-write

**Status**

mandatory



### ipRouteMetric5

An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

#### Syntax

INTEGER

#### Access

read-write

#### Status

mandatory

### ipRouteInfo

A reference to MIB definitions specific to the particular routing protocol which is responsible for this route, as determined by the value specified in the route's ipRouteProto value. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntactically valid object identifier. Any conformant implementation of ASN.1 and BER must be able to generate and recognize this value.

#### Syntax

OBJECT IDENTIFIER

#### Access

read-only

#### Status

mandatory

## The IP Address Translation Table

The IP address translation table contain the IpAddress to 'physical' address equivalences. Some interfaces do not use translation tables for determining address equivalences (for example, DDN-X.25 has an algorithmic method). If all interfaces are of this type, then the Address Translation table is empty, that is, has zero entries.

### ipNetToMediaIfIndex

The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

## Syntax

INTEGER

## Access

read-write

## Status

mandatory

## ipNetToMediaPhysAddress

The media-dependent ‘physical’ address.

## Syntax

PhysAddress

## Access

read-write

## Status

mandatory

## ipNetToMediaNetAddress

The IpAddress corresponding to the media-dependent ‘physical’ address.

## Syntax

IpAddress

## Access

read-write

## Status

mandatory

## ipNetToMediaType

The type of mapping. Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipNetToMediaTable. That is, it effectively disassociates the interface identified with the entry from the mapping identified with the entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipNetToMediaType object.

**Syntax**

INTEGER

**Access**

read-write

**Status**

mandatory

**Additional IP Objects**

Following are the additional IP objects.

**ipRoutingDiscards**

The number of routing entries which were chosen to be discarded even though they are valid. One possible reason for discarding such an entry could be to free-up buffer space for other routing entries.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**The ICMP Group**

Implementation of the ICMP group is mandatory for all systems.

**icmpInMsgs**

The total number of ICMP messages received by the entity. This counter includes all those counted by icmpInErrors.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

---

## icmpInErrors

The number of ICMP messages received by the entity but were determined as having ICMP-specific errors (such as, bad ICMP checksums, bad length).

### Syntax

Counter

### Access

read-only

### Status

mandatory

## icmpInDestUnreachs

The number of ICMP Destination Unreachable messages received.

### Syntax

Counter

### Access

read-only

### Status

mandatory

## icmpInTimeExcds

The number of ICMP Time Exceeded messages received.

### Syntax

Counter

### Access

read-only

### Status

mandatory

## icmpInParmProbs

The number of ICMP Parameter Problem messages received.

### Syntax

Counter

**Access**

read-only

**Status**

mandatory

**icmplnSrcQuenchs**

The number of ICMP Source Quench messages received.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmplnRedirects**

The number of ICMP Redirect messages received.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmplnEchos**

The number of ICMP Echo (request) messages received.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmplnEchoReps**

The number of ICMP Echo Reply messages received.

---

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpInTimestamps**

The number of ICMP Timestamp (request) messages received.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpInTimestampReps**

The number of ICMP Timestamp Reply messages received.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpInAddrMasks**

The number of ICMP Address Mask Request messages received.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpInAddrMaskReps**

The number of ICMP Address Mask Reply messages received.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpOutMsgs**

The total number of ICMP messages which this entity attempted to send. This counter includes all those counted by icmpOutErrors.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpOutErrors**

The number of ICMP messages which this entity did not send due to problems discovered within ICMP, such as a lack of buffers. This value should not include errors discovered outside the ICMP layer such as the inability of IP to route the resultant datagram. In some implementations, there may be no types of errors which contribute to this counter's value.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpOutDestUnreachs**

The number of ICMP Destination Unreachable messages sent.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpOutTimeExcds**

The number of ICMP Time Exceeded messages sent.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpOutParmProbs**

The number of ICMP Parameter Problem messages sent.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpOutSrcQuenches**

The number of ICMP Source Quench messages sent.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory



**icmpOutRedirects**

The number of ICMP Redirect messages sent. For a host, this object will always be zero, since hosts do not send redirects.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpOutEchos**

The number of ICMP Echo (request) messages sent.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpOutEchoReps**

The number of ICMP Echo Reply messages sent.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpOutTimestamps**

The number of ICMP Timestamp (request) messages sent.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpOutTimestampReps**

The number of ICMP Timestamp Reply messages sent.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpOutAddrMasks**

The number of ICMP Address Mask Request messages sent.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**icmpOutAddrMaskReps**

The number of ICMP Address Mask Reply messages sent.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

## The TCP Group

Implementation of the TCP group is mandatory for all systems that implement the TCP. Instances of object types that represent information about a particular TCP connection are transient; they persist only as long as the connection in question.

### tcpRtoAlgorithm

The algorithm used to determine the timeout value used for retransmitting unacknowledged octets.

#### Syntax

INTEGER

#### Access

read-only

#### Status

mandatory

### tcpRtoMin

The minimum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the LBOUND quantity described in RFC 793.

#### Syntax

INTEGER

#### Access

read-only

#### Status

mandatory

### tcpRtoMax

The maximum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the UBOUND quantity described in RFC 793.

#### Syntax

INTEGER

**Access**

read-only

**Status**

mandatory

**tcpMaxConn**

The limit on the total number of TCP connections the entity can support. In entities where the maximum number of connections is dynamic, this object should contain the value -1.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**tcpActiveOpens**

The number of times TCP connections have made a direct transition to the SYN-SENT state from the CLOSED state.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**tcpPassiveOpens**

The number of times TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**tcpAttemptFails**

The number of times TCP connections have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition to the LISTEN state from the SYN-RCVD state.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**tcpEstabResets**

The number of times TCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**tcpCurrEstab**

The number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT.

**Syntax**

Gauge

**Access**

read-only

**Status**

mandatory

## The TCP Connection Table

---

### tcpInSegs

The total number of segments received, including those received in error. This count includes segments received on currently established connections.

#### Syntax

Counter

#### Access

read-only

#### Status

mandatory

### tcpOutSegs

The total number of segments sent including those on current connections, but excluding those containing only retransmitted octets.

#### Syntax

Counter

#### Access

read-only

#### Status

mandatory

### tcpRetransSegs

The total number of segments retransmitted. That is, the number of TCP segments transmitted containing one or more previously transmitted octets.

#### Syntax

Counter

#### Access

read-only

#### Status

mandatory

## The TCP Connection Table

The TCP connection table contains information about this entity's existing TCP connections.

**tcpConnState**

The state of this TCP connection. The only value which may be set by a management station is deleteTCB(12). Accordingly, it is appropriate for an agent to return a “badValue” response if a management station attempts to set this object to any other value.

If a management station sets this object to the value deleteTCB(12), then this has the effect of deleting the TCB (as defined in RFC 793) of the corresponding connection on the managed Node. The result is an immediate termination of the connection.

**Syntax**

INTEGER

**Access**

read-write

**Status**

mandatory

**tcpConnLocalAddress**

The local IP address for this TCP connection. In the case of a connection in the listen state which is willing to accept connections for any IP interface associated with the Node, the value 0.0.0.0 is used.

**Syntax**

IpAddress

**Access**

read-only

**Status**

mandatory

**tcpConnLocalPort**

The local port number for this TCP connection.

**Syntax**

INTEGER (0..65535)

**Access**

read-only

## Additional TCP Objects

---

### Status

mandatory

### tcpConnRemAddress

The remote IP address for this TCP connection.

### Syntax

IpAddress

### Access

read-only

### Status

mandatory

### tcpConnRemPort

The remote port number for this TCP connection.

### Syntax

INTEGER (0..65535)

### Access

read-only

### Status

mandatory

## Additional TCP Objects

Following are the additional TCP objects.

### tcpInErrs

The total number of segments received in error (for example, bad TCP checksums).

### Syntax

Counter

### Access

read-only

### Status

mandatory



**tcpOutRsts**

The number of TCP segments sent containing the RST flag.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**The UDP Group**

Implementation of the UDP group is mandatory for all systems which implement the UDP.

**udpInDatagrams**

The total number of UDP datagrams delivered to UDP users.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**udpNoPorts**

The total number of received UDP datagrams for which there was no application at the destination port.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**udpInErrors**

The number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port.

## The UDP Listener Table

---

### Syntax

Counter

### Access

read-only

### Status

mandatory

## udpOutDatagrams

The total number of UDP datagrams sent from this entity.

### Syntax

Counter

### Access

read-only

### Status

mandatory

## The UDP Listener Table

The UDP listener table contains information about this entity's UDP end-points on which a local application is currently accepting datagrams.

## udpLocalAddress

The local IP address for this UDP listener. In the case of a UDP listener which is willing to accept datagrams for any IP interface associated with the Node, the value 0.0.0.0 is used.

### Syntax

IpAddress

### Access

read-only

### Status

mandatory

## udpLocalPort

The local port number for this UDP listener.

### Syntax

INTEGER (0..65535)

**Access**

read-only

**Status**

mandatory

**The EGP Group**

Implementation of the EGP group is mandatory for all systems which implement the EGP.

**egpInMsgs**

The number of EGP messages received without error.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**egpInErrors**

The number of EGP messages received that proved to be in error.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**egpOutMsgs**

The total number of locally generated EGP messages.

**Syntax**

Counter

**Access**

read-only

## The EGP Neighbor Table

---

### Status

mandatory

### egpOutErrors

The number of locally generated EGP messages not sent due to resource limitations within an EGP entity.

### Syntax

Counter

### Access

read-only

### Status

mandatory

## The EGP Neighbor Table

The EGP neighbor table contains information about this entity's EGP neighbors.

### egpNeighState

The EGP state of the local system with respect to this entry's EGP neighbor. Each EGP state is represented by a value that is one greater than the numerical value associated with the state in RFC 904.

### Syntax

INTEGER

### Access

read-only

### Status

mandatory

### egpNeighAddr

The IP address of this entry's EGP neighbor.

### Syntax

IpAddress

### Access

read-only

### Status

mandatory

**egpNeighAs**

The autonomous system of this EGP peer. Zero should be specified if the autonomous system number of the neighbor is not yet known.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**egpNeighInMsgs**

The number of EGP messages received without error from this EGP peer.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**egpNeighInErrs**

The number of EGP messages received from this EGP peer that proved to be in error (for example, bad EGP checksum).

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**egpNeighOutMsgs**

The number of locally generated EGP messages to this EGP peer.

**Syntax**

Counter

## The EGP Neighbor Table

---

### Access

read-only

### Status

mandatory

## egpNeighOutErrs

The number of locally generated EGP messages not sent to this EGP peer due to resource limitations within an EGP entity.

### Syntax

Counter

### Access

read-only

### Status

mandatory

## egpNeighInErrMsgs

The number of EGP-defined error messages received from this EGP peer.

### Syntax

Counter

### Access

read-only

### Status

mandatory

## egpNeighOutErrMsgs

The number of EGP-defined error messages sent to this EGP peer.

### Syntax

Counter

### Access

read-only

### Status

mandatory

**egpNeighStateUps**

The number of EGP state transitions to the UP state with this EGP peer.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**egpNeighStateDowns**

The number of EGP state transitions from the UP state to any other state with this EGP peer.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**egpNeighIntervalHello**

The interval between EGP Hello command retransmissions, in hundredths of a second. This represents the t1 timer as defined in RFC 904.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**egpNeighIntervalPoll**

The interval between EGP poll command retransmissions, in hundredths of a second. This represents the t3 timer as defined in RFC 904.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**egpNeighMode**

The polling mode of this EGP entity, either passive or active.

**Syntax**

INTEGER { active(1), passive(2) }

**Access**

read-only

**Status**

mandatory

**egpNeighEventTrigger**

A control variable used to trigger operator-initiated Start and Stop events. When read, this variable always returns the most recent value that egpNeighEventTrigger was set to. If it has not been set since the last initialization of the network management subsystem on the Node, it returns a value of “stop”.

When set, this variable causes a Start or Stop event on the specified neighbor, as specified on pages 8-10 of RFC 904. Briefly, a Start event causes an Idle peer to begin neighbor acquisition and a non-Idle peer to reinitiate neighbor acquisition. A stop event causes a non-Idle peer to return to the Idle state until a Start event occurs, either by egpNeighEventTrigger or otherwise.

**Syntax**

INTEGER { start(1), stop(2) }

**Access**

read-write

**Status**

mandatory

**egpAs**

The autonomous system number of this EGP entity.

**Syntax**

INTEGER



**Access**

read-only

**Status**

mandatory

## The Transmission Group

Based on the transmission media underlying each interface on a system, the corresponding portion of the Transmission group is mandatory for that system.

When Internet-standard definitions for managing transmission media are defined, the transmission group is used to provide a prefix for the names of those objects.

Typically, such definitions reside in the experimental portion of the MIB until they are "proven", then as a part of the Internet standardization process, the definitions are accordingly elevated and a new object identifier, under the transmission group is defined. By convention, the name assigned is:

type OBJECT IDENTIFIER ::= { transmission number }.

Where "type" is the symbolic value used for the media in the ifType column of the ifTable object, and "number" is the actual integer value corresponding to the symbol.

## The SNMP Group

Implementation of the SNMP group is mandatory for all systems which support an SNMP protocol entity. Some of the objects defined below will be zero-valued in those SNMP implementations that are optimized to support only those functions specific to either a management agent or a management station. In particular, it should be observed that the objects below refer to an SNMP entity, and there may be several SNMP entities residing on a managed Node.

### snmplnPmts

The total number of messages delivered to the SNMP entity from the transport service.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

---

## **snmpOutPkts**

The total number of SNMP messages which were passed from the SNMP protocol entity to the transport service.

### **Syntax**

Counter

### **Access**

read-only

### **Status**

mandatory

## **snmplnBadVersions**

The total number of SNMP messages which were delivered to the SNMP protocol entity and were for an unsupported SNMP version.

### **Syntax**

Counter

### **Access**

read-only

### **Status**

mandatory

## **snmplnBadCommunityNames**

The total number of SNMP messages delivered to the SNMP protocol entity which used a SNMP community name not known to the entity.

### **Syntax**

Counter

### **Access**

read-only

### **Status**

mandatory

## **snmplnBadCommunityUses**

The total number of SNMP messages delivered to the SNMP protocol entity which represented an SNMP operation which was not allowed by the SNMP community named in the message.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpInASNParseErrs**

The total number of ASN.1 or BER errors encountered by the SNMP protocol entity when decoding received SNMP messages.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpInTooBigs**

The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is “tooBig”.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpInNoSuchNames**

The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is "NoSuchName".

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmplnBadValues**

The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is “badValue”.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmplnReadOnlys**

The total number valid SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is “readOnly”. It should be noted that it is a protocol error to generate an SNMP PDU which contains the value “readOnly” in the error-status field, as such, this object is provided as a means of detecting incorrect implementations of the SNMP.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmplnGenErrs**

The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is “genErr”.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpInTotalReqVars**

The total number of MIB objects which have been retrieved successfully by the SNMP protocol entity as the result of receiving valid SNMP Get-Request and Get-Next PDUs.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpInTotalSetVars**

The total number of MIB objects which have been altered successfully by the SNMP protocol entity as the result of receiving valid SNMP Set-Request PDUs.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpInGetRequests**

The total number of SNMP Get-Request PDUs which have been accepted and processed by the SNMP protocol entity.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpInGetNexts**

The total number of SNMP Get-Next PDUs which have been accepted and processed by the SNMP protocol entity.

---

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpInSetRequests**

The total number of SNMP Set-Request PDUs which have been accepted and processed by the SNMP protocol entity.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpInGetResponses**

The total number of SNMP Get-Response PDUs which have been accepted and processed by the SNMP protocol entity.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpInTraps**

The total number of SNMP Trap PDUs which have been accepted and processed by the SNMP protocol entity.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpOutTooBig**

The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is “tooBig”

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpOutNoSuchNames**

The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status is "NoSuchName".

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpOutBadValues**

The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is “badValue”.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

---

## **snmpOutGenErrs**

The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is “genErr”.

### **Syntax**

Counter

### **Access**

read-only

### **Status**

mandatory

## **snmpOutGetRequests**

The total number of SNMP Get-Request PDUs which have been generated by the SNMP protocol entity.

### **Syntax**

Counter

### **Access**

read-only

### **Status**

mandatory

## **snmpOutGetNexts**

The total number of SNMP Get-Next PDUs which have been generated by the SNMP protocol entity.

### **Syntax**

Counter

### **Access**

read-only

### **Status**

mandatory

## **snmpOutSetRequests**

The total number of SNMP Set-Request PDUs which have been generated by the SNMP protocol entity.



**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpOutGetResponses**

The total number of SNMP Get-Response PDUs which have been generated by the SNMP protocol entity.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpOutTraps**

The total number of SNMP Trap PDUs which have been generated by the SNMP protocol entity.

**Syntax**

Counter

**Access**

read-only

**Status**

mandatory

**snmpEnableAuthenTraps**

Indicates whether the SNMP agent process is permitted to generate authentication-failure traps. The value of this object overrides any configuration information; as such, it provides a means whereby all authentication-failure traps may be disabled.

It is strongly recommended that this object be stored in non-volatile memory so that it remains constant between re-initializations of the network management system.

**Syntax**

INTEGER { enabled(1), disabled(2) }

**Access**

read-write

**Status**

mandatory

**Return Value**

Read returns enabled (1) if AuthFailureTrap = True, otherwise disabled (2).  
Writes not supported.

# Section 4

## Fibre Alliance MIB Objects

This section covers the implementation details for the Fibre Alliance Management Information Bases (FA-MIB) on the SANbox2 switch.

### FA MIB Definitions

The FA-MIB is a collection of structured objects that resides on the workstation with the manager application. These objects define the syntax for information exchanged between the manager and the agent. The textual substitutions in [Table 4-1](#) are specific to the FA-MIB and can be used in place of primitive data types.

**Table 4-1 FA-MIB Textual Substitutions**

Description	Syntax
FcNameId	OCTET STRING (SIZE(8))
FcGlobalId	OCTET STRING (SIZE(16))
FcAddressId	OCTET STRING (SIZE(3))
FcEventSeverity	INTEGER{ unknown (1), emergency (2), alert (3), critical (4), error (5), warning (6), notify (7), info (8), debug (9), mark (10) - All messages logged }

**Table 4-1 FA-MIB Textual Substitutions**

Description	Syntax
FcUnitType	<pre> INTEGER {     unknown(1)     other(2) - none of the following     hub(3) - passive connectivity unit supporting loop protocol.     switch(4) - active connectivity unit supporting multiple     protocols.     gateway(5) - unit that converts not only the interface but also     encapsulates the frame into another protocol. The assumption is     that there is always two gateways connected together. For     example, FC &lt;-&gt; ATM.     converter(6) - unit that converts from one interface to another.     For example, FC &lt;-&gt; SCSI.     hba(7) - host bus adapter     proxy-agent(8) - software proxy-agent     storage-device(9) - disk, cd, tape, etc.     host(10) - host computer     storage-subsystem(11) - raid, library, etc.     module(12) - subcomponent of a system     swdriver(13) - software driver     storage-access-device(14) - Provides storage management and     access for heterogeneous hosts and heterogeneous devices     wdm(15) - waveform division multiplexer     ups(16) - uninterruptable power supply }</pre>

## revisionNumber

The revision number for this MIB. The format of the revision value is as follows:

- (0) = high order major revision number
- (1) = low order major revision number
- (2) = high order minor revision number
- (3) = low order minor revision number

The value will be stored as an ASCII value. The following is the current value of 03.00 for this object.

- (0) = '0'
- (1) = '3'
- (2) = '0'

- (3) = '0'

**Syntax**

DisplayString (SIZE (4))

**Access**

read-only

**Status**

mandatory

**Return Value**

A four digit ASCII value (for example, 0300 for MIB revision 3.0).

## Connectivity Unit Group

The objects described in this section are not in a table format. An example of how to access one of these objects is:

"snmpget localhost public fcmgmt.connSet.uNumber.0".

### uNumber

The number of connectivity units present on this system (represented by this agent). May be a count of the boards in a chassis or the number of full boxes in a rack.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**Return Value**

The number of switches in fabric.

### systemURL

The top-level URL of the system. If it does not exist, the value is an empty string. The URL format is implementation dependant and can have keywords embedded that are preceded by a percent sign (for example, %USER).

**Syntax**

DisplayString

**Access**

read-write

**Status**

mandatory

**Return Value**

The switch IP address. For example, http://10.0.0.1. Writes not supported, returns "NoSuchName".

**statusChangeTime**

The sysuptime timestamp at which the last status change occurred for any members of the set, in centiseconds.

**Syntax**

TimeTicks

**Access**

read only

**Status**

obsolete

**Return Value**

This object is obsolete. Always returns error status "NoSuchName".

**configurationChangeTime**

The sysuptime timestamp at which the last configuration change occurred for any members of the set, in centiseconds. This represents a union of change information for connUnitConfigurationChangeTime.

**Syntax**

TimeTicks

**Access**

read only

**Status**

obsolete

**Return Value**

This object is obsolete. Always returns error status "NoSuchName".

## connUnitTableChangeTime

The sysuptime timestamp at which the connUnitTable was updated (an entry was either added or deleted), in centiseconds.

### Syntax

TimeTicks

### Access

read only

### Status

obsolete

### Return Value

This object is obsolete. Always returns error status "NoSuchName".

## Connectivity Table

The objects described in this section are in a table format indexed by switch World Wide Name. An example of how to access one of these objects given a WWN of 100000c0dd0090a7 is:

```
"snmpget localhost public fcmgmt.connSet.connUnit-  
Table.connUnitEntry.connUnitId..16.0.0.192.221.0.144.167.0.0.0.0.0.0.0".
```

## connUnitId

The unique identification for this connectivity unit among those within this proxy domain. The value must be unique within the proxy domain because it is the index variable for connUnitTable. The value assigned to a given connectivity unit should be persistent across agent and unit resets. It should be the same as connUnitGlobalId if connUnitGlobalId is known and stable.

### Syntax

FcGlobalId

### Access

read-only

### Status

mandatory

### Return Value

The World Wide Name of the switch. For example, 10 00 00 C0 DD 00 71 C9.

---

**connUnitGlobalId**

An optional global-scope identifier for this connectivity unit. It must be a WWN for this connectivity unit or 16 octets of value zero.

**Syntax**

connUnitGlobalId

**Access**

read-only

**Status**

mandatory

**Return Value**

The World Wide Name of the switch. For example, 10 00 00 C0 DD 00 71 C9.

**connUnitType**

The type of this connectivity unit.

**Syntax**

FcUnitType

**Access**

read-only

**Status**

mandatory

**Return Value**

switch (4)

**connUnitNumports**

Number of physical ports in the connectivity unit (internal/embedded, external).

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory



**Return Value**

The number of ports on the switch.

**connUnitState**

The operational state of the switch mapped.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**Return Value**

Refer to [Table 4-2](#) for switch operational states.

**Table 4-2 Switch Operational States**

Switch State	Return State
online	online (2)
offline	offline (3)
diagnostic	offline (3)
other	unknown (1)

**connUnitStatus**

Overall status of the connectivity unit. The goal of this object is to be the single poll point to check the status of the connunit. If there is any other component that has warning, then this should be set to warning.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**Return Value**

Refer to [Table 4-3](#) for connectivity unit return values. Return value will be OK (3), unless one or more of the following occurs.

**Table 4-3 Connectivity Unit Return Values**

Status	Return Value
If either power supply failed	failed (5)
If one cooling fan failed	warning (4)
If two cooling fans failed	failed (5)
If temperature status = "Warm"	warning (4)
If temperature status = "Overheating"	failed (5)
If any port down	warning (4)
If OperState = Downed	failed (5)

**connUnitProduct**

The sml attribute Oper.Switch.Type (for example, SANbox2).

**Syntax**

DisplayString (SIZE (0..79))

**Access**

read-only

**Status**

mandatory

**Return Value**

SANbox2-16

**connUnitSn**

The serial number for this connectivity unit.

**Syntax**

DisplayString (SIZE (0..79))

**Access**

read-only

**Status**

mandatory

**Return Value**

The chassis serial number.

**connUnitUpTime**

The number of centiseconds since the last unit initialization.

**Syntax**

TimeTicks

**Access**

read-only

**Status**

mandatory

**Return Value**

The time since last POST was executed. For example, 3 days 22 hours.

**connUnitUrl**

URL to launch a management application, if applicable. Otherwise, it's an empty string. In a standalone unit, this would be the same as the top-level URL. This has the same definition as systemURL for keywords. If write is not supported, then the return value is invalid. This value will be retained across boots.

**Syntax**

DisplayString

**Access**

read-write

**Status**

mandatory

**Return Value**

The switch IP address. For example, http://10.0.0.1. Writes not supported, returns "NoSuchName".

**connUnitDomainId**

24 bit Fibre Channel address ID of this connectivity unit, right justified with leading zeros if required. This should be set to the Fibre Channel address ID, or if it

is a switch, it would be set to the Domain Controller address. If this value is not applicable, return all bits set to one.

**Syntax**

OCTET STRING (SIZE(3))

**Access**

read-only

**Status**

mandatory

**Return Value**

The domain controller address. For example, FF FC 65.

**connUnitProxyMaster**

A value of “yes” means this is the proxy master unit for a set of managed units. For example, this could be the only unit with a management card in it for a set of units. A standalone unit should return “yes” for this object.

**Syntax**

```
INTEGER {  
    unknown(1),  
    no(2),  
    yes(3)  
}
```

**Access**

read-only

**Status**

mandatory

**Return Value**

Always returns yes (3).

**connUnitPrincipal**

Whether this connectivity unit is the principal unit within the group of fabric elements. If this value is not applicable, the return is unknown.

**Syntax**

```
INTEGER {  
    unknown(1),  
    no(2),
```

```
yes(3)
}
```

**Access**

read-only

**Status**

mandatory

**Return Value**

For the principal switch, returns yes (3); otherwise returns no (2).

**connUnitNumSensors**

Number of sensors in the connUnitSensorTable elements. If this value is not applicable, return unknown.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**Return Value**

Returns 6 (for example, 2 fans, 2 power supplies, and 2 temperature sensors).

**connUnitStatusChangeTime**

The sysuptime timestamp, in centiseconds, at which the last status change occurred.

**Syntax**

TimeTicks

**Access**

read-only

**Status**

obsolete

**Return Value**

This object is obsolete. Always returns error status "NoSuchName".

---

**connUnitConfigurationChangeTime**

The sysuptime timestamp, in centiseconds, at which the last configuration change occurred.

**Syntax**

TimeTicks

**Access**

read-only

**Status**

obsolete

**Return Value**

This object is obsolete. Always returns error status "NoSuchName".

**connUnitNumRevs**

The number of revisions in the connUnitRevsTable.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**Return Value**

The number of entries in the revision table (for example, 4). The revision table contains the revision numbers of all components of the switch.

**connUnitNumZones**

Number of zones defined in connUnitZoneTable.

**Syntax**

INTEGER

**Access**

read-only

**Status**

obsolete

**Return Value**

This object is obsolete. Always returns error status "NoSuchName".

**connUnitModuleId**

This is a unique ID, persistent between boots, that can be used to group a set of connUnits together into a module. The intended use would be to create a connUnit with a connUnitType of "module" to represent a physical or logical group of connectivity units. Then, the value of the group would be set to the value of connUnitId for this "container" connUnit. connUnitModuleId should be zeros if this connUnit is not part of a module.

**Syntax**

FcGlobalId

**Access**

read-only

**Status**

mandatory

**Return Value**

The World Wide Name of the switch. For example, 10 00 00 C0 DD 00 71 C9.

**connUnitName**

A display string containing a name for this connectivity unit. This object value should be persistent between boots.

**Syntax**

DisplayString (SIZE(0..79))

**Access**

read-write

**Status**

mandatory

**Return Value**

The SymbolicName of switch. The default is: SANbox2. Writes are supported.

**connUnitInfo**

A display string containing information about this connectivity unit. This object value should be persistent between boots.

**Syntax**

DisplayString

**Access**

read-write

**Status**

mandatory

**Return Value**

The switch Description field. The default is: QLogic SANbox2 FC Switch.  
Writes are supported.

**connUnitControl**

This object is used to control the addressed connUnit. “Cold Start” and “Warm Start” are as defined in MIB-II and are not meant to be a factory reset.

- resetConnUnitColdStart: the addressed unit performs a “Cold Start” reset.
- resetConnUnitWarmStart: the addressed unit performs a “Warm Start” reset.
- offlineConnUnit: the addressed unit puts itself into an implementation dependant “offline” state. In general, if a unit is in an offline state, it cannot be used to perform meaningful Fibre Channel work.
- onlineConnUnit: the addressed unit puts itself into an implementation dependant “online” state. In general, if a unit is in an online state, it is capable of performing meaningful Fibre Channel work.

Each implementation may chose not to allow any or all of these values on a SET.

**Syntax**

```
INTEGER {  
    unknown(1),  
    invalid(2),  
    resetConnUnitColdStart(3),  
    resetConnUnitWarmStart(4),  
    offlineConnUnit(5),  
    onlineConnUnit(6)  
}
```

**Access**

read-write



**Status**

mandatory

**Return Value**

Refer to the following tables for connUnitControl values.

**Table 4-4 connUnitControl Read Return Values**

Switch Setting	Return Value
Online	Online (6)
Offline	Offline (5)
Diagnostic	Offline (5)
Other	Unknown (1)

**Table 4-5 connUnitControl Write Control Values**

Control Value	Result
Cold Reset (3)	Reset
Offline (5)	Offline
Online (6)	Online
other	Not supported

**connUnitContact**

Contact information for this connectivity unit, and is persistent across boots.

**Syntax**

DisplayString (SIZE (0..79))

**Access**

read-write

**Status**

mandatory

**Return Value**

The default is: <sysContact undefined>

**connUnitLocation**

Location information for this connectivity unit, and is persistent across boots.

**Syntax**

DisplayString (SIZE (0..79))

**Access**

read-write

**Status**

mandatory

**Return Value**

The default is: <sysLocation undefined>

**connUnitEventFilter**

This value defines the event severity that will be logged by this connectivity unit. All events of severity less than or equal to connUnitEventFilter are logged in connUnitEventTable.

**Syntax**

FcEventSeverity

**Access**

read-write

**Status**

mandatory

**Return Value**

The switch log level setting. Refer to the following tables for connUnitEventFilter values.

**Table 4-6 connUnitEventFilter Read Return Values**

Severity Levels	Return Value
Critical	Critical (4)
Warn	Warning (6)
Info	Info (8)
None	Unknown (1)

**Table 4-7 connUnitEventFilter Control Write Values**

Control Value	Result
Emergency (2)	Critical
Alert (3)	Critical
Critical (4)	Critical
Error (5)	Warn
Warning (6)	Warn
Notify (7)	Info
Info (8)	Info
Debug (9)	Info
Mark (10)	Info
Unknown (1)	None

**connUnitNumEvents**

Number of events currently in the connUnitEventTable.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**Return Value**

An integer indicating the number of events in the event table.

**connUnitMaxEvents**

Maximum number of events that can be defined in connUnitEventTable.

**Syntax**

INTEGER

**Access**

read-only

## Revision Table

---

### Status

mandatory

### Return Value

Always returns 30.

## connUnitEventCurrID

The last used event ID (connUnitEventIndex).

### Syntax

INTEGER

### Access

read-only

### Status

mandatory

### Return Value

The event ID of the last alarm.

## Revision Table

The objects described in this section are in a table format indexed by World Wide Name and Index. The Table of revisions for hardware and software elements. There are four revision items in each switch. An example of how to access one of these objects given a WWN of 100000c0dd0090a7 is:

```
"snmpget localhost public fcmgmt.connSet.connUnitRevsTable.connUnitRevsEntry.connUnitRevsUnitId.16.0.0.192.221.0.144.167.0.0.0.0.0.0.1".
```

## connUnitRevsUnitId

The connUnitId of the connectivity unit that contains this revision table.

### Syntax

FcGlobalId

### Access

read-only

### Status

mandatory

### Return Value

The World Wide Name of the switch. For example, 10 00 00 C0 DD 00 71 C9.

## Revision Table

---

### connUnitRevsIndex

A unique value among all connUnitRevsEntrys with the same value of connUnitRevsUnitId, in the range between 1 and connUnitNumRevs[connUnitRevsUnitId].

#### Syntax

INTEGER (1..2147483647)

#### Access

read-only

#### Status

mandatory

#### Return Value

The revision table index.

### connUnitRevsRevId

A vendor-specific string identifying a revision of a component of the connUnit indexed by connUnitRevsUnitId.

#### Syntax

DisplayString

#### Access

read-only

#### Status

mandatory

#### Return Value

Refer to [Table 4-8](#) for connUnitRevsRevId return values.

**Table 4-8 ConnUnitRevsRevId Return Values**

Table Index	Return Value
1	Hardware ASIC Version
2	Hardware PCB Version
3	Flasher Shell Version
4	Active Firmware Version

## Sensor Table

---

### connUnitRevsDescription

Description of a component to which the revision corresponds.

#### Syntax

DisplayString

#### Access

read-only

#### Status

mandatory

#### Return Value

Refer to [Table 4-9](#) for connUnitRevsDescription return values.

**Table 4-9 ConnUnitRevsDescription Return Values**

Table Index	Return Value
1	Hardware ASIC Version
2	Hardware PCB Version
3	Flasher Shell Version
4	Active Firmware Version

## Sensor Table

The objects described in this section are in a table format indexed by World Wide Name and Index. The Index is the sensor number being interrogated. There are six sensor items in each switch. An example of how to access one of these objects given a WWN of 100000c0dd0090a7 is:

```
"snmpget localhost public fcmgmt.connSet.connUnitSensorTable.connUnitSensorEntry.connUnitSensorUnitId.16.0.0.192.221.0.144.167.0.0.0.0.0.0.1"
```

### connUnitSensorUnitId

The connUnitId of the connectivity unit that contains this sensor table.

#### Syntax

FcGlobalId

#### Access

read-only

## Sensor Table

---

### Status

mandatory

### Return Value

The World Wide Name of the switch. For example, 10 00 00 C0 DD 00 71 C9.

## connUnitSensorIndex

A unique value among all connUnitSensorEntrys with the same value of connUnitSensorUnitId, in the range between 1 and connUnitNumSensor[connUnitSensorUnitId].

### Syntax

INTEGER (1..2147483647)

### Access

read-only

### Status

mandatory

### Return Value

The sensor table index.

## connUnitSensorName

A textual identification of the sensor intended primarily for operator use.

### Syntax

DisplayString

### Access

read-only

### Status

mandatory

### Return Value

Refer to [Table 4-10](#) for connUnitSensorName return values.

**Table 4-10 ConnUnitSensorName Return Values**

Table Index	Return Value
1	Board temperature sensor #1
2	Board temperature sensor #2

**Table 4-10 ConnUnitSensorName Return Values**

Table Index	Return Value
3	Fan #1 status
4	Fan #1 status
5	Power supply #1 status
6	Power supply #2 status

**connUnitSensorStatus**

The status indicated by the sensor.

**Syntax**

```
INTEGER {  
    unknown(1)  
    other(2) - the sensor indicates other than ok (warning or failure).  
    ok(3) - the sensor indicates ok  
    warning(4) - the sensor indicates a warning  
    failed(5) - the sensor indicates failure  
}
```

**Access**

read-only

**Status**

mandatory

**Return Value**

Refer to the following tables for connUnitSensorStatus return values.

**Table 4-11 ConnUnitSensorStatus Return Values for Board Temperature**

Switch Value	Return Value
Normal	OK (3)
Warm	Warning (4)
Overheating	Warning (4)
Other	Unknown (1)



**Table 4-12 ConnUnitSensorStatus Return Values for Fan Status**

Switch Value	Return Value
Good	OK (3)
Bad	Failed (5)
Other	Unknown (1)

**Table 4-13 ConnUnitSensorStatus Return Values for Power Supply Status**

Switch Value	Return Value
Good	OK (3)
Bad	Failed (5)
Other	Unknown (1)

**connUnitSensorInfo**

Miscellaneous static information about the sensor, such as its serial number.

**Syntax**

DisplayString

**Access**

read-only

**Status**

mandatory

**Return Value**

Always returns an empty string.

**connUnitSensorMessage**

This describes the status of the sensor as a message. It may also provide more resolution on the sensor indication. For example, “Cover temperature 1503K, above nominal operating range” ::= { connUnitSensorEntry 6 }.

**Syntax**

DisplayString

**Access**

read-only

**Status**

mandatory

**Return Value**

Refer to [Table 4-14](#) for connUnitSensorMessage values.

**Table 4-14 ConnUnitSensorMessage Values**

Sensor	Value
Temperature	Degrees in C
Fan	Good/Bad
Power Supply	Good/Bad

**connUnitSensorType**

The type of component being monitored by this sensor.

**Syntax**

```
INTEGER {  
    unknown(1),  
    other(2),  
    battery(3),  
    fan(4),  
    power-supply(5),  
    transmitter(6),  
    enclosure(7),  
    board(8),  
    receiver(9)  
}
```

**Access**

read-only

**Status**

mandatory

**Return Value**

Refer to [Table 4-15](#) for connUnitSensorType return values.

**Table 4-15 ConnUnitSensorType Return Values**

Sensor	Value
Temperature	Board (8)
Fan	Fan (4)
Power Supply	Power Supply (5)

**connUnitSensorCharacteristic**

The characteristics being monitored by this sensor.

**Syntax**

```
INTEGER {  
    unknown(1),  
    other(2),  
    temperature(3),  
    pressure(4),  
    emf(5),  
    currentValue(6), - current is a keyword  
    airflow(7),  
    frequency(8),  
    power(9),  
    door(10)  
}
```

**Access**

read-only

**Status**

mandatory

**Return Value**

Refer to [Table 4-16](#) for connUnitSensorCharacteristic values.

**Table 4-16 ConnUnitSensorCharacteristic Values**

Sensor	Value
Temperature	Temperature (3)
Fan	Airflow (7)

## Port Table

**Table 4-16 ConnUnitSensorCharacteristic Values**

Sensor	Value
Power Supply	Power (9)

## Port Table

The objects described in this section are in a table format indexed by World Wide Name and Index. The Index is the port number being interrogated. There may be different numbers of ports in each switch so the agent must determine the maximum allowable index on a switch by switch basis. An example of how to access one of these objects given a WWN of 100000c0dd0090a7 is:

```
"snmpget localhost public fcmgmt.connSet.connUnitPortTable.connUnitPortEntry.connUnitPortUnitId.16.0.0.192.221.0.144.167.0.0.0.0.0.0.0.1"
```

### connUnitPortUnitId

The connUnitId of the connectivity unit that contains this port.

#### Syntax

FcGlobalId

#### Access

read-only

#### Status

mandatory

#### Return Value

The World Wide Name of the switch. For example, 10 00 00 C0 DD 00 71 C9.

### connUnitPortIndex

A unique value among all connUnitPortEntrys on this connectivity unit, between 1 and connUnitNumPort[connUnitPortUnitId].

#### Syntax

INTEGER (1..2147483647)

#### Access

read-only

#### Status

mandatory

**Return Value**

The index for each port on the switch.

**connUnitPortType**

The port type.

**Syntax**

```
INTEGER {  
    unknown(1),  
    other(2),  
    not-present(3),  
    hub-port(4),  
    n-port(5), - end port for fabric  
    nl-port(6), - end port for loop  
    fl-port(7), - public loop  
    f-port(8), - fabric port  
    e-port(9), - fabric expansion port  
    g-port(10), - generic fabric port  
    domain-ctl(11), - domain controller  
    hub-controller(12),  
    scsi(13), - parallel SCSI port  
    escon(14),  
    lan(15),  
    wan(16),  
    ac(17), - AC power line  
    dc(18), - DC power line  
    ssa(19) - serial storage architecture  
}
```

**Access**

read-only

**Status**

mandatory

**Return Value**

Refer to [Table 4-17](#) for connUnitPortType return values.

**Table 4-17 ConnUnitPortType Return Values**

Switch Port Type	Return Value
G	g-port (10)
FL	fl-port (7)
F	f-port (8)
E	e-port (9)
TL	other (2)
Donor	other (2)
other	unknown (1)

**connUnitPortFCClassCap**

Bit mask that specifies the classes of service capability of this port. If this is not applicable, return all bits set to zero.

The bits have the following definition:

unknown - 0

class-f - 1

class-one - 2

class-two - 4

class-three - 8

class-four - 16

class-five - 32

class-six - 64

**Syntax**

OCTET STRING (SIZE (2))

**Access**

read-only

**Status**

mandatory

**Return Value**

Always returns 0x0d (Class f, Class 2, and Class 3).

---

## connUnitPortFCClassOp

Bit mask that specifies the classes of service that are currently operational. If this is not applicable, return all bits set to zero. This object has the same definition as connUnitPortFCClassCap" ::= { connUnitPortEntry 5 }.

### Syntax

OCTET STRING (SIZE (2))

### Access

read-only

### Status

mandatory

### Return Value

If F or FL, returns 0x0c (Class 2, and Class 3), else returns 0x0d (Class f, Class 2, and Class 3).

## connUnitPortState

The user selected state of the port hardware.

### Syntax

```
INTEGER {  
  unknown(1),  
  online(2), - available for meaningful work  
  offline(3), - not available for meaningful work  
  bypassed(4), - no longer used (4/12/00)  
  diagnostics(5)  
}
```

### Access

read-only

### Status

mandatory

### Return Value

Refer to [Table 4-18](#) for connUnitPortState return values.

**Table 4-18 ConnUnitPortState Return Values**

Port Value	Return Value
Online	online (2)
Offline	offline (3)
Downed	offline (3)
Diagnostic	diagnostics (5)
other	unknown (1)

### connUnitPortStatus

An overall protocol status for the port. This value of connUnitPortState is not online, then this is reported Unknown.

#### Syntax

```
INTEGER {  
    unknown(1),  
    unused(2), - device cannot report this status  
    ready(3), - FCAL Loop or FCPH Link reset protocol; initialization complete  
    warning(4), - do not use (4/12/00)  
    failure(5), - do not use (4/12/00)  
    notparticipating(6), - loop not participating and does not have a loop address  
    initializing(7), - protocol is proceeding  
    bypass(8), - do not use (4/12/00)  
    ols(9) - FCP offline status  
}
```

#### Access

read-only

#### Status

mandatory

#### Return Value

Always returns unused (2).

### connUnitPortTransmitterType

The technology of the port transceiver.



**Syntax**

```
INTEGER {  
    unknown(1),  
    other(2),  
    unused(3),  
    shortwave(4),  
    longwave(5),  
    copper(6),  
    scsi(7),  
    longwaveNoOFC(8),  
    shortwaveNoOFC(9),  
    longwaveLED(10),  
    ssa(11)  
}
```

**Access**

read-only

**Status**

mandatory

**Return Value**

Refer to [Table 4-19](#) for connUnitPortTransmitterType return values.

**Table 4-19 ConnUnitPortTransmitterType Return Values**

SFP Transmitter Type	Return Value
Not Installed	Unused (3)
SL	Shortwave (4)
LL	Longwave (5)
LC	LongwaveNoOFC (8)
SN	ShortwaveNoOFC (9)
Other	Unknown (1)

**connUnitPortModuleType**

The module type of the port connector.

**Syntax**

```
INTEGER {  
    unknown(1),  
    other(2),  
    gbic(3),  
    embedded(4), - fixed (oneXnine)  
    glm(5),  
    gbicSerialId(6),  
    gbicNoSerialId(7),  
    gbicNotInstalled(8),  
    smallFormFactor(9) - this is generically a small form factor connector.  
}
```

**Access**

read-only

**Status**

mandatory

**Return Value**

Always returns smallFormFactor(9).

**connUnitPortWwn**

The World Wide Name of the port, if applicable, otherwise returns all zeros.

**Syntax**

```
FcNameId
```

**Access**

read-only

**Status**

mandatory

**Return Value**

Port World Wide Name. For example, the return value for port #2 would be 20 02 00 C0 DD 00 71 C9, and the return value for port #14 would be 20 0E 00 C0 DD 00 71 C9.

**connUnitPortFCId**

This is the assigned Fibre Channel ID of this port. This value is expected to be a Big Endian value of 24 bits. If this is a loop, then it is the ALPA that is connected.

## Port Table

---

If this is an E-Port, then it will only contain the domain ID left justified, zero filled. If this port does not have a Fibre Channel address, returns all bits set to 1.

### Syntax

FcAddressId

### Access

read-only

### Status

mandatory

### Return Value

The address for each port based on Domain, Area, and ALPA. For example, port #15 would be equal to 640F00 (Domain = 0x64, Area = 0x0F, ALPA = 0x00).

## connUnitPortSn

The serial number of the unit. If not applicable, returns an empty string.

### Syntax

DisplayString (SIZE(0..79))

### Access

read-only

### Status

mandatory

### Return Value

Always returns an empty string.

## connUnitPortRevision

The port revision. For example, for a GBIC.

### Syntax

DisplayString (SIZE(0..79))

### Access

read-only

### Status

mandatory

**Return Value**

Always returns an empty string.

**connUnitPortVendor**

The port vendor. For example, for a GBIC.

**Syntax**

DisplayString (SIZE(0..79))

**Access**

read-only

**Status**

mandatory

**Return Value**

Always returns an empty string.

**connUnitPortSpeed**

The speed of the port in kilobytes per second.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**Return Value**

The operational speed, otherwise returns the administrative speed setting. If 1 Gbps, returns 106250 (0x19F0A). If 2 Gbps, returns 212500 (0x33E14).

**connUnitPortControl**

This object is used to control the addressed connUnit's port.

- **resetConnUnitPort:** If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific "reset" operation. Examples of these operations are: the Link Reset protocol, the Loop Initialization protocol, or a re-synchronization occurring between the transceiver in the addressed port to the transceiver that the port is connected to.

- **bypassConnUnitPort:** If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific “bypass” operation. Examples of these operations are transitioning from online to offline, a request (NON-PARTICIPATING) command to the Loop Port state machine, or removal of the port from an arbitrated loop by a hub.
- **unbypassConnUnitPort:** If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific “unbypass” operation. Examples of these operations are the Link Failure protocol, a request (PARTICIPATING) command to the Loop Port state machine, or addition of the port to an arbitrated loop by a hub.
- **offlineConnUnitPort:** If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific “offline” operation. Examples of these operations are disabling a port's transceiver, the Link Failure protocol, request (NON-PARTICIPATING) command to the Loop Port state machine, or removal of the port from an arbitrated loop by a hub.
- **onlineConnUnitPort:** If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific “online” operation. Examples of these operations are enabling a port's transceiver, the Link Failure protocol, request (PARTICIPATING) command to the Loop Port state machine, or addition of the port from an arbitrated loop by a hub.
- **resetConnUnitPortCounters:** If the addressed connUnit allows this operation to be performed to this port, the addressed port statistics table counters will be set to zero.

Each implementation may choose not to allow any or all of these values on a SET. On a read, if you do not support write, then return invalid. Otherwise, return the last control operation attempted.

### Syntax

```
INTEGER {  
    unknown(1),  
    invalid(2),  
    resetConnUnitPort(3),  
    bypassConnUnitPort(4),  
    unbypassConnUnitPort(5),  
    offlineConnUnitPort(6),  
    onlineConnUnitPort(7),  
    resetConnUnitPortCounters(8)
```

```
}
```

**Access**

read-write

**Status**

mandatory

**Return Value**Refer to [Table 4-20](#) for connUnitPortControl read return values.**Table 4-20 ConnUnitPortControl Read Return Values**

Port Value	Return Value
Online	online (7)
Offline	offline (6)
Diagnostic	offline (6)
other	unknown (1)

Refer to [Table 4-21](#) for connUnitPortControl write command values.**Table 4-21 ConnUnitPortControl Write Command Values**

Control Value	Command Sent
Online (3)	online
Offline (6)	offline
ResetCounters (8)	clear counters
other	error returned

**connUnitPortName**

A user-defined name for this port. This means that up to DisplayString characters may be supported. If less than, then the name will be truncated in the connunit.

**Syntax**

INTEGER

**Access**

read-write

**Status**

mandatory

**Return Value**

The symbolic port name. For example, port #2 would return “Port2” (Port# is the default return value format).

**connUnitPortPhysicalNumber**

This is the internal port number this port is known by. In many implementations, this should be the same as connUnitPortIndex. Some implementations may have an internal port representation not compatible with the rules for table indexes. In that case, provide the internal representation of this port in this object. This value may also be used in the connUnitLinkPortNumberX or connUnitLinkPortNumberY objects of the connUnitLinkTable.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**Return Value**

The physical port number.

**connUnitPortStatObject**

This contains the OID of the first object of the table that contains the statistics for this particular port. If this has a value of zero, then there are no statistics available for this port. The port type information will help identify the statistics objects that will be found in the table.

**Syntax**

OBJECT IDENTIFIER

**Access**

read-only

**Status**

deprecated

**Return Value**

The port object ID for QLogic SANbox2 (1.3.6.1.3.94.4.5.1.1).

---

**connUnitPortProtocolCap**

Bit mask that specifies the driver level protocol capability of this port. If this is not applicable, returns all bits set to zero.

The bits have the following definitions:

unknown - 0

Loop - 1

Fabric - 2

SCSI - 4

TCP/IP - 8

VI - 16

FICON - 32

**Syntax**

OCTET STRING (SIZE (2))

**Access**

read-only

**Status**

mandatory

**Return Value**

Always returns 0x03 (Loop, Fabric).

**connUnitPortProtocolOp**

Bit mask that specifies the driver level protocol(s) that are currently operational. If not applicable, return all bits set to zero. This object has the same definition as connUnitPortProtocolCap.

**Syntax**

OCTET STRING (SIZE (2))

**Access**

read-only

**Status**

mandatory

**Return Value**

Always returns 0x00 (Unknown).



---

## connUnitPortNodeWwn

The Node World Wide Name of the port if applicable, otherwise all zeros. This should have the same value for a group of related ports. The container is defined as the largest physical entity. For example, all ports on HBAs on a host will have the same Node WWN. All ports on the same storage subsystem will have the same Node WWN." ::= { connUnitPortEntry 22 }.

### Syntax

FcNameId

### Access

read-only

### Status

mandatory

### Return Value

The World Wide Name of the switch. For example, 10 00 00 C0 DD 00 71 C9.

## connUnitPortHWState

The hardware detected state of the port.

### Syntax

```
INTEGER {  
    unknown(1),  
    failed(2), - port failed diagnostics  
    bypassed(3), - FCAL bypass, loop only  
    active(4), - connected to a device  
    loopback(5), - Port in external loopback  
    txfault(6), - Transmitter fault  
    noMedia(7), - media not installed linkDown  
    (8) - waiting for activity (rx sync)  
}
```

### Access

read-only

### Status

mandatory

### Return Value

Refer to [Table 4-22](#) for connUnitPortHWState port state return values.

**Table 4-22 ConnUnitPortHWState Port State Return Values**

Port State	Return Value
If OperState = Downed	Failed (2)
If GBIC = Not Installed	NoMedia (7)
If SyncStatus = SyncAcquired	Active (4)
If SyncStatus = SyncLost	LinkDown (8)
Other	Unknown (1)

## Event Table

The objects described in this section are in a table format indexed by World Wide Name and Index. The maximum index is determined based on the number of events in the table. An example of how to access one of these objects given a WWN of 100000c0dd0090a7 is:

```
"snmpget localhost public fcmgmt.connSet.connUnitEventTable.connUnitEventEntry.connUnitEventUnitId.16.0.0.192.221.0.144.167.0.0.0.0.0.0.1".
```

### connUnitEventUnitId

The connUnitId of the connectivity unit that contains this event table.

#### Syntax

FcGlobalId

#### Access

read-only

#### Status

mandatory

#### Return Value

The World Wide Name of the switch. For example, 10 00 00 C0 DD 00 71 C9 00 00 00 00 00 00 00 00.

### connUnitEventIndex

Each connectivity unit has its own event buffer. As it wraps, it may write over previous events. This object is an index into the buffer. It is recommended that this table be read using getNext's to retrieve the initial table. The management application should read the event table at periodic intervals and then determine if any new entries were added by comparing the last known index value with the current highest index value. The management application should then update its copy of

## Event Table

---

the event table. If the read interval is too long, it is possible that there may be events that may not be contained in the agent's internal event buffer. For example, an agent may read events 50-75. At the next read interval, connUnitEventCurrID is 189. If the management application tries to read event index 76, and the agent's internal buffer is 100 entries max, event index 76 will no longer be available.

The index value is an incrementing integer starting from one every time there is a table reset. On table reset, all contents are emptied and all indexes are set to zero. When an event is added to the table, the event is assigned the next higher integer value than the last item entered into the table. If the index value reaches its maximum value, the next item entered will cause the index value to roll over and start at one again.

### Syntax

INTEGER (1..2147483647)

### Access

read-only

### Status

mandatory

### Return Value

The table index.

## connUnitEventId

The internal event ID. Incremented for each event, ranging between 1 and connUnitMaxEvents. Not used as table index to simplify the agent implementation. When this reaches the end of the range specified by connUnitMaxEvents, the ID will roll over to start at one. This value will be set back to one at reset. The relationship of this value to the index is that internal event ID may represent a smaller number than a 32 bit integer (for example, maximum 100 entries) and would only have a value range up to connUnitMaxEvents.

### Syntax

INTEGER

### Access

read-only

### Status

deprecated

**Return Value**

Unsupported. Always returns error status "NoSuchName".

**connUnitREventTime**

The real time when the event occurred. It has the following format.

DDMMYYYY HHMMSS

DD=day number

MM=month number

YYYY=year number

HH=hour number

MM=minute number

SS=seconds number

If not applicable, return either a NULL string or "00000000 000000".

**Syntax**

DisplayString (SIZE (0..15))

**Access**

read-only

**Status**

mandatory

**Return Value**

The timestamp of the event.

**connUnitSEventTime**

This is the sysuptime timestamp when the event occurred.

**Syntax**

connUnitSEventTime

**Access**

read-only

**Status**

mandatory

**Return Value**

Always returns error status "NoSuchName".

**connUnitEventSeverity**

The event severity level.

**Syntax**

FcEventSeverity

**Access**

read-only

**Status**

mandatory

**Return Value**

Always returns error status "NoSuchName".

**connUnitEventType**

The type of this event.

**Syntax**

```
INTEGER {  
    unknown(1),  
    other(2),  
    status(3),  
    configuration(4),  
    topology(5)  
}
```

**Access**

read-only

**Status**

mandatory

**Return Value**

Always returns 3 (Status).

**connUnitEventObject**

This is used with the connUnitEventType to identify which object the event refers to. Examples include connUnitPortStatus.connUnitId.connUnitPortIndex and connUnitStatus.connUnitId.

**Syntax**

OBJECT IDENTIFIER

**Access**

read-only

## Link Table

---

### Status

mandatory

### Return Value

Always returns error status "NoSuchName".

## connUnitEventDescr

The description of the event.

### Syntax

DisplayString

### Access

read-only

### Status

mandatory

### Return Value

The event description in the form: "[Id][times-tamp][severity][module][Description]"

## Link Table

The objects described in this section are in a table format indexed by World Wide Name and Index. The index is an index into the link table for the switch. There may be as many link entries as there are ports. An example of how to access one of these objects given a WWN of 100000c0dd0090a7 is:

```
"snmpget localhost public fcmgmt.connSet.connUnitLink-
Table.connUnitLinkEntry.connUnitLinkUnitId.16.0.0.192.221.0.144.167.0.0.0.0.0
.0.0.0.1".
```

If the agent is able to discover links which do not directly attach to members of its agency and its discovery algorithm gives some assurance the links are recently valid, it MAY include these links. Link information entered by administrative action MAY be included even if not validated directly if the link has at least one endpoint in this agency, but SHOULD NOT be included otherwise.

A connectivity unit should fill the table in as best it can. One of the methods to fill this in would be to use the RNID ELS (ANSI document 99-422v0). This allows one to query a port for the information needed for the link table.

This table is accessed either directly if the management software has an index value or via GetNexts. The value of the indexes are not required to be contiguous. Each entry created in this table will be assigned an index. This relationship is kept

persistent until the entry is removed from the table or the system is reset. The total number of entries are defined by the size of the table

### **connUnitLinkId**

The connUnitId of the connectivity unit that contains this link table.

#### **Syntax**

connUnitLinkId

#### **Access**

read-only

#### **Status**

mandatory

#### **Return Value**

The World Wide Name of the switch. For example, 10 00 00 C0 DD 00 71 C9 00 00 00 00 00 00 00.

### **connUnitLinkIndex**

This index is used to create a unique value for each entry in the link table with the same connUnitLinkId. The value can only be reused if it is not currently in use and the value is the next candidate to be used. This value wraps at the highest value represented by the size of INTEGER. This value is reset to zero when the system is reset and the first value to be used is one.

#### **Syntax**

INTEGER (1..2147483647)

#### **Access**

read-only

#### **Status**

mandatory

#### **Return Value**

The table index.

### **connUnitLinkNodeIdx**

The Node WWN of the unit at one end of the link. If the Node WWN is unknown and the Node is a connUnit in the responding agent then the value of this object must be equal to its connUnitID.

**Syntax**

OCTET STRING (SIZE(16))

**Access**

read-only

**Status**

mandatory

**Return Value**

The World Wide Name of the local switch for each entry in the link table. For example, 10 00 00 C0 DD 00 71 C9 00 00 00 00 00 00 00.

**connUnitLinkPortNumberX**

The port number on the unit specified by connUnitLinkNodeIdx if known, otherwise -1. If the value is non-negative, then it will be equal to connUnitPortPhysicalNumber.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**Return Value**

The local port number for each entry in the link table.

**connUnitLinkPortWwnX**

The port WWN of the unit specified by connUnitLinkNodeIdx if known, otherwise 16 octets of binary 0" ::= { connUnitLinkEntry 5 }.

**Syntax**

connUnitLinkPortWwnX

**Access**

read-only

**Status**

mandatory



**Return Value**

The local World Wide port number for each entry in the link table.

**connUnitLinkNodeIdY**

The Node WWN of the unit at the other end of the link. If the Node WWN is unknown and the Node is a connUnit in the responding SNMP agency, then the value of this object must be equal to its connUnitID.

**Syntax**

OCTET STRING (SIZE(16))

**Access**

read-only

**Status**

mandatory

**Return Value**

The remote World Wide Node number for each entry in the link table.

**connUnitLinkPortNumberY**

The port number on the unit specified by connUnitLinkNodeIdY if known, otherwise -1. If the value is non-negative, then it will be equal to connUnitPortPhysical-Number.

**Syntax**

OCTET STRING (SIZE(16))

**Access**

read-only

**Status**

mandatory

**Return Value**

The remote port number for inter-switch link, if known. Otherwise, -1 (0xFFFFFFFF).

**connUnitLinkPortWwnY**

The port WWN on the unit specified by connUnitLinkNodeIdY if known, otherwise 16 octets of binary 0" ::= { connUnitLinkEntry 8 }.

**Syntax**

FcGlobalId

## Link Table

---

### Access

read-only

### Status

mandatory

### Return Value

The remote Port World Wide Name for each entry in the link table, if known.

## connUnitLinkAgentAddressY

The address of an FCMGMT MIB agent for the Node identified by connUnitLinkNodeidY, if known. Otherwise 16 octets of binary 0" ::= {connUnitLinkEntry 9}.

### Syntax

OCTET STRING (SIZE(16))

### Access

read-only

### Status

mandatory

### Return Value

The remote IP address of the remote switch, if known. Otherwise, returns sixteen zeroes.

## connUnitLinkAgentAddressTypeY

If connUnitLinkAgentAddressY is nonzero, it is a protocol address. ConnUnitLinkAgentAddressTypeY is the "address family number" assigned by IANA to identify the address format.

### Syntax

INTEGER

### Access

read-only

### Status

mandatory

### Return Value

Always returns 1 (Ipv4).

---

**connUnitLinkAgentPortY**

The IP port number for the agent. This is provided in case the agent is at a non-standard SNMP port.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**Return Value**

The remote port number for inter-switch link, if known. Otherwise, returns -1 (0xFFFFFFFF).

**connUnitLinkUnitTypeY**

Type of the Fibre Channel connectivity unit as defined in connUnitType.

**Syntax**

FcUnitType

**Access**

read-only

**Status**

mandatory

**Return Value**

The type of remote device in the link table. For example, switch (4).

**connUnitLinkConnIdY**

This is the Fibre Channel ID of this port. If the connectivity unit is a switch, this is expected to be a Big Endian value of 24 bits. If this is loop, then it is the ALPA that is connected. If this is an E\_Port, then it will only contain the domain ID. If not any of those, unknown or cascaded loop, returns all bits set to 1.

**Syntax**

OCTET STRING (SIZE(3))

**Access**

read-only

## Port Statistics Table

---

### Status

mandatory

### Return Value

The remote Fibre Channel address of each entry in the link table.

## connUnitLinkCurrIndex

The last used link index.

### Syntax

INTEGER

### Access

read-only

### Status

mandatory

### Return Value

The last used link table index number.

## Port Statistics Table

The objects described in this section are in a table format indexed by World Wide Name and Index. The index represents the port number to interrogate. An example of how to access one of these objects given a WWN of 100000c0dd0090a7 is:

"snmpget localhost public fcmgmt.statSet.connUnitPortStatTable.connUnitPortStatEntry.connUnitPortStatUnitId.16.0.0.192.221.0.144.167.0.0.0.0.0.0.1".

There is one and only one statistics table for each individual port. For all objects in statistics table, if the object is not supported by the conn unit then the high order bit is set to 1 with all other bits set to zero. The high order bit is reserved to indicate if the object is supported or not. All objects start at a value of zero at hardware initialization and continue incrementing till end of 63 bits and then wrap to zero.

## connUnitPortStatUnitId

The connUnitId of the connectivity unit that contains this port statistic table.

### Syntax

FcGlobalId

### Access

read-only

**Status**

mandatory

**Return Value**

The World Wide Name of the switch. For example, 10 00 00 C0 DD 00 71 C9.

**connUnitPortStatIndex**

A unique value among all entries in this table, between 0 and connUnitNumPort[connUnitPortUnitId].

**Syntax**

INTEGER (0..2147483647)

**Access**

read-only

**Status**

mandatory

**Return Value**

The port table index.

**connUnitPortStatCountError**

A count of the errors that have occurred on this port.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

A hexadecimal value indicating the total number of errors for a port.

**connUnitPortStatCountTxObjects**

The number of frames/packets/IOs/etc transmitted by this port. A Fibre Channel frame starts with SOF and ends with EOF. Fibre Channel loop devices should not count frames passed through. This value represents the sum total for all other Tx objects.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

A hexadecimal value indicating the total number of frames transmitted by a port.

**connUnitPortStatCountRxObjects**

The number of frames/packets/IOs/etc received by this port. A Fibre Channel frame starts with SOF and ends with EOF. Fibre Channel loop devices should not count frames passed through. This value represents the sum total for all other Rx objects.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

A hexadecimal value indicating the total number of frames received by a port.

**connUnitPortStatCountTxElements**

The number of octets or bytes that have been transmitted by this port. One second periodic polling of the port. This value is saved and compared with the next polled value to compute net throughput. For Fibre Channel, ordered sets are not included in the count.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

## Port Statistics Table

---

### Status

mandatory

### Return Value

A hexadecimal value indicating the total number of words transmitted by a port.

## connUnitPortStatCountRxElements

The number of octets or bytes that have been received by this port. One second periodic polling of the port. This value is saved and compared with the next polled value to compute net throughput. For Fibre Channel, ordered sets are not included in the count.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

A hexadecimal value indicating the total number of words received by a port.

## connUnitPortStatCountBBCreditZero

Count of transitions in/out of BBCredit zero state. The other side is not providing any credit. This is a Fibre Channel statistic only.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

---

**connUnitPortStatCountInputBuffersFull**

Count of occurrences when all input buffers of a port were full and outbound buffer-to-buffer credit transitioned to zero. There is no credit to provide to other side. This is a Fibre Channel statistic only.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

**connUnitPortStatCountFBSYFrames**

Count of times that FBSY was returned to this port as a result of a frame that could not be delivered to the other end of the link. This occurs if either the fabric or the destination port is temporarily busy. Port can only occur on SOFc1 frames (the frames that establish a connection). This is a Fibre Channel only statistic. This is the sum of all classes. If you cannot keep the by-class counters, then keep the sum counters.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

**connUnitPortStatCountPBSYFrames**

Count of times that PBSY was returned to this port as a result of a frame that could not be delivered to the other end of the link. This occurs if the destination port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). This is a Fibre Channel only statistic. This is the sum of all classes. If you cannot keep the by class counters, then keep the sum counters.



## Port Statistics Table

---

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

0Unsupported. Always returns high bit set: 0x800000000.

## connUnitPortStatCountFRJTFrames

Count of times that FRJT was returned to this port as a result of a frame that was rejected by the fabric. This is the total for all classes and is a Fibre Channel only statistic.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

## connUnitPortStatCountPRJTFrames

Count of times that FRJT was returned to this port as a result of a frame that was rejected at the destination N\_Port. This is the total for all classes and is a Fibre Channel only statistic.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

**Return Value**

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

**connUnitPortStatCountClass1RxFrames**

Count of Class 1 frames received at this port. This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

**connUnitPortStatCountClass1TxFrames**

Count of Class 1 frames transmitted out this port. This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

**connUnitPortStatCountClass1FBSYFrames**

Count of times that FBSY was returned to this port as a result of a Class 1 frame that could not be delivered to the other end of the link. This occurs if either the fabric or the destination port is temporarily busy. FBSY can only occur on SOFc1 frames (the frames that establish a connection). This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

## Port Statistics Table

---

### Access

read-only

### Status

mandatory

### Return Value

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

## **connUnitPortStatCountClass1PBSYFrames**

Count of times that PBSY was returned to this port as a result of a Class 1 frame that could not be delivered to the other end of the link. This occurs if the destination N\_Port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). This is a Fibre Channel only statistic.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

## **connUnitPortStatCountClass1FRJTFrames**

Count of times that FRJT was returned to this port as a result of a Class 1 frame that was rejected by the fabric. This is a Fibre Channel only statistic.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

**connUnitPortStatCountClass1PRJTFrames**

Count of times that FRJT was returned to this port as a result of a Class 1 frame that was rejected at the destination N\_Port. This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

**connUnitPortStatCountClass2RxFrames**

Count of Class 2 frames received at this port. This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

The total number of Class 2 frames received by a port.

**connUnitPortStatCountClass2TxFrames**

Count of Class 2 frames transmitted out this port. This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

### Return Value

The total number of Class 2 frames transmitted by a port.

### **connUnitPortStatCountClass2FBSYFrames**

Count of times that FBSY was returned to this port as a result of a Class 2 frame that could not be delivered to the other end of the link. This occurs if either the fabric or the destination port is temporarily busy. FBSY can only occur on SOFc1 frames (the frames that establish a connection). This is a Fibre Channel only statistic.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

### **connUnitPortStatCountClass2PBSYFrames**

Count of times that PBSY was returned to this port as a result of a Class 2 frame that could not be delivered to the other end of the link. This occurs if the destination N\_Port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). This is a Fibre Channel only statistic.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

### **connUnitPortStatCountClass2FRJTFrames**

Count of times that FRJT was returned to this port as a result of a Class 2 frame that was rejected by the fabric. This is a Fibre Channel only statistic.

## Port Statistics Table

---

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

## **connUnitPortStatCountClass2PRJTFrames**

Count of times that FRJT was returned to this port as a result of a Class 2 frame that was rejected at the destination N\_Port. This is a Fibre Channel only statistic.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

## **connUnitPortStatCountClass3RxFrames**

Count of Class 3 frames received at this port. This is a Fibre Channel only statistic.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

The total number of Class 3 frames received by a port.

---

**connUnitPortStatCountClass3TxFrames**

Count of Class 3 frames transmitted out this port. This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

The total number of Class 3 frames transmitted by a port.

**connUnitPortStatCountClass3Discards**

Count of Class 3 frames that were discarded upon reception at this port. There is no FBSY or FRJT generated for Class 3 frames. They are simply discarded if they cannot be delivered. This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

The total number of Class3Toss frames for a port.

**connUnitPortStatCountRxMulticastObjects**

Count of Multicast frames or packets received at this port.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

### Return Value

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

### **connUnitPortStatCountTxMulticastObjects**

Count of Multicast frames or packets transmitted out this port.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

### **connUnitPortStatCountRxBroadcastObjects**

Count of Broadcast frames or packets received at this port.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

### **connUnitPortStatCountTxBroadcastObjects**

Count of Broadcast frames or packets transmitted out this port. On a Fibre Channel loop, count only OPN<sub>r</sub> frames generated.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only



**Status**

mandatory

**Return Value**

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

**connUnitPortStatCountRxLinkResets**

Count of link resets. This is the number of LR's received. This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

The total number of RxLinkResets received by a port.

**connUnitPortStatCountTxLinkResets**

Count of link resets. The number of LR's transmitted. This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

The total number of RxLinkResets transmitted by a port.

**connUnitPortStatCountNumberLinkResets**

Count of link resets and LIPs detected at this port. The number times the reset link protocol is initiated. These are the count of the logical resets, and a count of the number of primitives. This is a Fibre Channel only statistic.

## Port Statistics Table

---

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

The total number of TotalLinkResets for a port.

## **connUnitPortStatCountRxOfflineSequences**

Count of offline primitive OLS's received at this port. This is a Fibre Channel only statistic.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

The total number of RxOfflineSeq's received by a port.

## **connUnitPortStatCountTxOfflineSequences**

Count of offline primitive OLS's transmitted by this port. This is a Fibre Channel only statistic.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

The total number of TxOfflineSeq's transmitted by a port.

### connUnitPortStatCountNumberOfflineSequences

Count of offline primitive sequences received at this port. This is a Fibre Channel only statistic.

#### Syntax

OCTET STRING (SIZE (8))

#### Access

read-only

#### Status

mandatory

#### Return Value

The total number of TotalOfflineSeq's received by a port.

### connUnitPortStatCountLinkFailures

Count of link failures. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). This is a Fibre Channel only statistic.

#### Syntax

OCTET STRING (SIZE (8))

#### Access

read-only

#### Status

mandatory

#### Return Value

The total number of LinkFailures's for a port.

### connUnitPortStatCountInvalidCRC

Count of frames received with invalid CRC. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Loop ports should not count CRC errors passing through when monitoring. This is a Fibre Channel only statistic.

#### Syntax

OCTET STRING (SIZE (8))

#### Access

read-only

## Port Statistics Table

---

### Status

mandatory

### Return Value

The total number of InvalidCRC's received by a port.

## **connUnitPortStatCountInvalidTxWords**

Count of invalid transmission words received at this port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). This is a Fibre Channel only statistic.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

## **connUnitPortStatCountPrimitiveSequenceProtocolErrors**

Count of primitive sequence protocol errors detected at this port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). This is a Fibre Channel only statistic.

### Syntax

OCTET STRING (SIZE (8))

### Access

read-only

### Status

mandatory

### Return Value

The total number of PrimSeqErrors for a port.

## **connUnitPortStatCountLossofSignal**

Count of instances of signal loss detected at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

**connUnitPortStatCountLossofSynchronization**

Count of instances of synchronization loss detected at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

The total number LossOfSync's detected by this port.

**connUnitPortStatCountInvalidOrderedSets**

Count of invalid ordered sets received at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

---

**connUnitPortStatCountFramesTooLong**

Count of frames received at this port where the frame length was greater than what was agreed to in FLOGI/PLOGI. This could be caused by losing the end of frame delimiter. This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

**connUnitPortStatCountFramesTruncated**

Count of frames received at this port where the frame length was less than the minimum indicated by the frame header (normally 24 bytes). It could be more if the DFCTL field indicates an optional header should have been present. This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

**connUnitPortStatCountAddressErrors**

Count of frames received with unknown addressing.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

The total number of InvalidDestAddr frames received by a port.

**connUnitPortStatCountDelimiterErrors**

Count of invalid frame delimiters received at this port. An example is a frame with a Class 2 start and a Class 3 at the end. This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

**connUnitPortStatCountEncodingDisparityErrors**

Count of disparity errors received at this port. This is a Fibre Channel only statistic.

**Syntax**

OCTET STRING (SIZE (8))

**Access**

read-only

**Status**

mandatory

**Return Value**

Unsupported. Always returns high order bit to 1 with all other bits set to zero.

**Simple Name Server Table**

The objects described in this section are in a table format indexed by World Wide Name and Index. The index represents the table index. An example of how to access one of these objects given a WWN of 100000c0dd0090a7 is:

## Simple Name Server Table

---

"snmpget localhost public

fcmgmt.connUnitServiceSet.connUnitServiceTables.connUnitSnsTable.connUnitSnsEntry.connUnitSnsId.16.0.0.192.221.0.144.167.0.0.0.0.0.0.1".

The Fibre Channel Simple Name Server table contains an entry for each device presently known to this connUnit. There will not be any version on this since FC-GS3 does not define a version today.

This table is accessed either directly if the management software has an index value or via GetNexts. The value of the indexes are not required to be contiguous. Each entry created in this table will be assigned an index. This relationship is kept persistent until the entry is removed from the table or the system is reset. The total number of entries are defined by the size of the table

### connUnitSnsId

The connUnitId of the connectivity unit that contains this Name Server table.

#### Syntax

OCTET STRING (SIZE (16))

#### Access

read-only

#### Status

mandatory

#### Return Value

The World Wide Name of the switch. For example, 10 00 00 C0 DD 00 71 C9.

### connUnitSnsPortIndex

The physical port number of this SNS table entry. Each physical port has an SNS table with 1-n entries indexed by ConnUnitSnsPortIdentifier (port address).

#### Syntax

INTEGER

#### Access

read-only

#### Status

mandatory

#### Return Value

The name server table index.



**connUnitSnsPortIdentifier**

The Port Identifier for this entry in the SNS table.

**Syntax**

FcAddressId

**Access**

read-only

**Status**

mandatory

**Return Value**

The 24-bit Fibre Channel address for each entry in the name server table based on Domain, Area, and ALPA.

**connUnitSnsPortName**

The Port World Wide Name for this entry in the SNS table.

**Syntax**

FcNameId

**Access**

read-only

**Status**

mandatory

**Return Value**

The Port World Wide Name of the device in the name server table.

**connUnitSnsNodeName**

The Node name for this entry in the SNS table.

**Syntax**

FcNameId

**Access**

read-only

**Status**

mandatory

### Return Value

The Node World Wide Name of the device in the name server table.

### connUnitSnsClassOfSvc

The classes of service offered by this entry in the SNS table.

#### Syntax

OCTET STRING (SIZE (1))

#### Access

read-only

#### Status

mandatory

### Return Value

A value indicating the first registered class of service for an entry in name server table.

### connUnitSnsNodeIPAddress

The IPv6 formatted address of the Node for this entry in the SNS table.

#### Syntax

OCTET STRING (SIZE (16))

#### Access

read-only

#### Status

mandatory

### Return Value

Unsupported. Always returns error status "NoSuchName".

### connUnitSnsProcAssoc

The process associator for this entry in the SNS table.

#### Syntax

OCTET STRING (SIZE (16))

#### Access

read-only

**Status**

mandatory

**Return Value**

Unsupported. Always returns error status "NoSuchName".

**connUnitSnsFC4Type**

The FC-4 types supported by this entry in the SNS table.

**Syntax**

OCTET STRING (SIZE (1))

**Access**

read-only

**Status**

mandatory

**Return Value**

A value indicating the FC4 types registered for the device in the name server table. For example: 04 (SCSI-FCP).

**connUnitSnsPortType**

The port type of this entry in the SNS table.

**Syntax**

OCTET STRING (SIZE (1))

**Access**

read-only

**Status**

mandatory

**Return Value**

A value indicating the PortType for the entry in the name server table. Refer to [Table 4-23](#) for connUnitPortType port type return values.

**Table 4-23 ConnUnitPortType State Return Values**

Port Type	Return Value (hexidecimal)
N	1
NL	2

**Table 4-23 ConnUnitPortType State Return Values**

Port Type	Return Value (hexidecimal)
F/NL	3
NX	7F
F	8
FL	82
E	84
B	85

**connUnitSnsPortIPAddress**

The IPv6 formatted address of this entry in the SNS table.

**Syntax**

OCTET STRING (SIZE (16))

**Access**

read-only

**Status**

mandatory

**Return Value**

Unsupported. Always returns error status "NoSuchName".

**connUnitSnsFabricPortName**

The fabric port name of this entry in the SNS table.

**Syntax**

FcNameId

**Access**

read-only

**Status**

mandatory

**Return Value**

The switch Port World Wide Name for the device in the name server table.

**connUnitSnsHardAddress**

The hard ALPA of this entry in the SNS table.

**Syntax**

FcAddressId

**Access**

read-only

**Status**

mandatory

**Return Value**

Unsupported. Always returns 0.

**connUnitSnsSymbolicPortName**

The symbolic port name of this entry in the SNS table.

**Syntax**

DisplayString (SIZE (0..79))

**Access**

read-only

**Status**

mandatory

**Return Value**

The symbolic Port Name registered by the device in the name server table.

**connUnitSnsSymbolicNodeName**

The symbolic Node name of this entry in the SNS table.

**Syntax**

DisplayString (SIZE (0..79))

**Access**

read-only

**Status**

mandatory

**Return Value**

The symbolic Node Name registered by the device in the name server table.

## Trap Table

Traps are asynchronous messages sent from the agent (residing on the switch) to the manager (residing on the workstation) to identify significant events.

There can be up to 5 trap addresses within the trap table. All trap information is stored within the switch and is accessible to Telnet and the SNMP agent, and is persistent between boots. An example of how to access one of these objects given an IP address of 10.32.165.4 is:

```
"snmpget localhost public fcmgmt.trapReg.trapRegTable.trapRegEntry.trapReg-Filter.10.32.165.4.162".
```

A trap event is reported when the incoming error has a severity level less than or equal to the configured severity level. The trap event types and trap severity levels are listed in [Table 4-24](#).

**Table 4-24 Trap Severity Levels**

Event Type	Severity Level
Unknown	1
Emergency	2
Alert	3
Critical	4
Error	5
Warning	6
Notify	7
Info	8
Debug	9
Mark	10

### trapMaxClients

The maximum number of SNMP trap recipients supported by the connectivity unit.

#### Syntax

INTEGER

#### Access

read-only

**Status**

mandatory

**Return Value**

Always returns 5.

**trapClientCount**

The current number of rows in the trap table.

**Syntax**

INTEGER

**Access**

read-only

**Status**

mandatory

**Return Value**

A value (1-5) indicating number of configured trap clients.

**trapRegIpAddress**

The IP address of a client registered for traps.

**Syntax**

IpAddress

**Access**

read-only

**Status**

mandatory

**Return Value**

The IP addresses (as defined in the trap table) of where to send traps when they occur.

**trapRegPort**

The UDP port to send traps to for this host. Normally this would be the standard trap port (162). This object is an index and must be specified to create a row in this table.

**Syntax**

INTEGER (1..2147483647)

## Trap Table

---

### Access

read-only

### Status

mandatory

### Return Value

The configured port number of where to send traps when they occur. The port number can be configured in the switch SNMP setup parameters.

## trapRegFilter

This value defines the trap severity filter for this trap host. The connUnit will send traps to this host that have a severity level less than or equal to this value. The default value of this object is “warning”.

### Syntax

FcEventSeverity

### Access

read-write

### Status

mandatory

### Return Value

A value indicating the trap severity level. Refer to [Table 4-24](#) for trap severity levels.

## trapRegRowState

Specifies the state of the row.

- rowDestroy
  - READ: Can never happen.
  - WRITE: Remove this row from the table.
- rowInactive
  - READ: Indicates that this row does exist, but that traps are not enabled to be sent to the target.
  - WRITE: If the row does not exist, and the agent allows writes to the trap table, then a new row is created. The values of the optional columns will be set to default values. Traps are not enabled to be sent to the target. If the row already existed, then traps are disabled from being sent to the target.



- rowActive
  - READ: Indicates that this row exists, and that traps are enabled to be sent to the target.
  - WRITE: If the row does not exist, and the agent allows writes to the trap table, then a new row is created. The values of the optional columns will be set to default values. Traps are enabled to be sent to the target. If the row already exists, then traps are enabled to be sent to the target.

A value of “rowActive” or “rowInactive” must be specified to create a row in the table.

### Syntax

```
INTEGER {
    rowDestroy(1), - Remove row from table.
    rowInactive(2), - Row exists, but traps disabled
    rowActive(3) - Row exists and is enabled for sending traps
}
```

### Access

read-write

### Status

mandatory

### Return Value

Returns rowActive (3), if valid entry in trap table.

## Related Traps

The following traps contain the trap information being sent from the agent to the manager.

### connUnitStatusChange

The overall status of the connectivity unit has changed. The recommended severity level (for filtering) is “alert”.

Variables: { connUnitStatus, connUnitState }

### connUnitDeletedTrap

A connUnit has been deleted from this agent. The recommended severity level (for filtering) is “warning”.

Variables: { connUnitId }

## Related Traps

### connUnitEventTrap

An event has been generated by the connectivity unit. The recommended severity level (for filtering) is “info”.

Variables:

{ connUnitEventId, connUnitEventType, connUnitEventObject, connUnitEventDescr }

Figure 4-1 provides the standard format of the connUnitEventDescr variable.

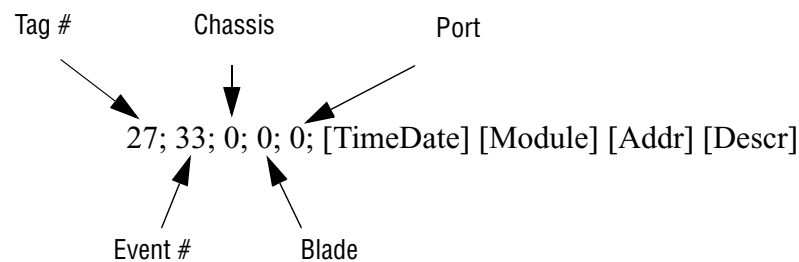


Figure 4-1 connUnitEventDescr Variable Format

Table 4-25 lists the fields in the connUnitEventDescr variable.

**Table 4-25 connUnitEventDescr Variable Field Descriptions**

connUnitEventDescr Variable	Description
Tag #	The number that identifies the event.
Event #	The event counter.
Chassis	The switch on which the event occurred.
Blade	The board on which the event occurred.
Port	The port on which the event occurred.
TimeDate	The time stamp of the event.
Module	The software module where the event was initiated.
Addr	The address in the software module where the event was initiated.
Descr	The description of the event.

Table 4-26 lists the possible trap strings returned for the connUnitEventDescr variable.

**Table 4-26 connUnitEventDescr Trap List**

ID	Notification	Error Level	connUnitEventDescr Examples
1	Snmp Config Change	info	1;2;0;0;0;[Tue Apr 30 14:03:06 2002][0][0x0][INFO: Snmp config change]
2	Switch Config Change	info	2;21;0;0;0;[Tue Apr 30 14:07:17 2002][0][0x0][INFO: Switch config change]
3	System Config Change	info	3;20;0;0;0;[Tue Apr 30 14:07:16 2002][0][0x0][INFO: System config change]
4	UserPort Config Change	info	4;37;0;0;15;[Tue Apr 30 14:07:22 2002][0][0x0][INFO: User Port config change]
5	Fabric Oper Change	info	5;82;0;0;0;[Tue Apr 30 15:17:15 2002][0][0x0][INFO: Fabric oper change]
6	Topology Oper Change	info	6;82;0;0;0;[Tue Apr 30 15:17:15 2002][0][0x0][INFO: Topology oper change]
7	Zoning Change	info	7;38;0;0;0;[Tue Apr 30 14:11:28 2002][0][0x0][INFO: Zoning change]
8	Zoning Merge Alarm	critical	8;38;0;0;0;[Tue Apr 30 14:11:28 2002][0][0x0][Zone Merge response indicates failure to merge, Isolating link]
9	NameServer Oper Change	info	9;101;0;0;0;[Tue Apr 30 15:7:15 2002][0][0x0][INFO: Nameserver oper change]
10	Eport Oper Change	info	10;77;0;0;0;[Tue Apr 30 15:17:10 2002][0][0x0][INFO: Eport oper change]
11	Eport Converge Alarm	critical	11;38;0;0;0;[Tue Apr 30 14:11:28 2002][0][0x0][Port 0 not configured for E-Port capability]

Table 4-26 connUnitEventDescr Trap List

ID	Notification	Error Level	connUnitEventDescr Examples
12	Switch Oper Change	info	12;40;0;0;0;[Tue Apr 30 14:15:54 2002][0][0x0][INFO: Switch oper change]
13	Switch State Change	info	13;41;0;0;0;[Tue Apr 30 14:15:55 2002][0][0x0][INFO: Switch state change]
14	Switch Access Alarm	warning	14;42;0;0;0;[Tue Apr 30 14:15:56 2002][0][0x0][Unknown user attempted to log into switch from 10.1.1.14]
15	Switch Admin Alarm	warning	15;43;0;0;0;[Tue Apr 30 14:15:57 2002][0][0x0][Admin access has timed out for a UI user Bob]
16	Switch Generic Alarm	critical	16;75;0;0;0;[Tue Apr 30 14:22:17.911 2002][A4101][Switch Management:0xdd009048.306.4][Admin mode for user admin@OB-session1 is was canceled by user admin@OB-session1]
17	Switch Reset Alarm	critical	17;104;0;0;0;[Tue Apr 30 15:37:26.500 2002][A4096][Switch Management:0xdd009048.306.4][The switch is being reset]
18	Chassis Oper Change	info	18;40;0;0;0;[Tue Apr 30 14:13:44 2002][0][0x0][INFO: Chassis oper change]
19	Chassis PS Bad Alarm	critical	19;1;0;0;0;[Tue Apr 30 14:01:44.118 2002][A8195][0xdd009048.296][WARNING: Pwr1 failure]
20	Chassis PS Ok Alarm	critical	20;1;0;0;0;[Tue Apr 30 14:01:44.118 2002][A8195][0xdd009048.296][ INFO: Pwr1 OK]
21	Chassis Fan Bad Alarm	critical	21;0;0;0;0;[Tue Apr 30 14:01:40.078 2002][A8194][0xdd009048.296][WARNING: Fan2 failure]

Table 4-26 connUnitEventDescr Trap List

ID	Notification	Error Level	connUnitEventDescr Examples
22	Chassis Fan Ok Alarm	critical	22;1;0;0;0;[Tue Apr 30 14:01:44.118 2002][A8195][0xdd009048.296][INFO: Fan2 OK]
23	Blade Oper Change	info	23;40;0;0;0;[Tue Apr 30 14:13:44 2002][0][0x0][INFO: Blade oper change]
24	Blade Overheat Alarm	critical	24;1;0;0;0;[Tue Apr 30 14:01:44.118 2002][A8195][0xdd009048.296][CRITICAL: Temp1 sensor reads 40C]
25	Blade Overwarm Alarm	critical	25;1;0;0;0;[Tue Apr 30 14:01:44.118 2002][A8195][0xdd009048.296][WARNING: Temp1 sensor reads 32C]
26	Blade Temp Ok Alarm	critical	26;1;0;0;0;[Tue Apr 30 14:01:44.118 2002][A8195][0xdd009048.296][INFO: Temp1 sensor has returned to normal range]
27	UserPort State Change	info	27;40;0;0;0;[Tue Apr 30 14:13:44 2002][0][0x0][INFO: User Port state change]
28	UserPort Oper Change	info	28;76;0;0;2;[Tue Apr 30 15:16:55 2002][0][0x0][INFO: User Port oper change]

### connUnitSensorStatusChange

The overall status of the connectivity unit has changed. The recommended severity level (for filtering) is “alert”.

Variables: { connUnitSensorStatus }

### connUnitPortStatusChange

The overall status of the connectivity unit has changed. The recommended severity level (for filtering) is “alert”.

Variables: { connUnitPortStatus, connUnitPortState }

### coldStart

A coldStart trap signifies that the SNMPv2 entity, acting in an agent role, is reinitializing itself and that its configuration may have been altered.

## Related Traps

---

### authenticationFailure

An authenticationFailure trap signifies that the SNMPv2 entity, acting in an agent role, has received a protocol message that is not properly authenticated. While all implementations of the SNMPv2 must be capable of generating this trap, the snmpEnableAuthenTraps object indicates whether this trap will be generated.

# Section 5

## Fabric Element MIB Objects

This section covers the implementation details for the Fabric Element Management Information Bases (FE-MIB) on the SANbox2 switch.

### Fibre Channel FE MIB Definitions

The textual substitutions in [Table 5-1](#) are specific to the FE-MIB and can be used in place of primitive data types.

**Table 5-1 FA-MIB Textual Substitutions**

Description	Syntax
MilliSeconds	Unsigned32
MicroSeconds	Unsigned32
FcNameId	OCTET STRING (SIZE (8))
FcAddressId	OCTET STRING (SIZE (3))
FcRxDataFieldSize	Integer32 (128..2112)
FcBbCredit	Integer32 (0..32767)
FcphVersion	Integer32 (0..255)
FcStackedConnMode	INTEGER { none(1), transparent(2), lockedDown(3) }
FcCosCap	BITS { classF(0), class1(1), class2(2), class3(3), class4(4), class5(5), class6(6) }

Table 5-1 FA-MIB Textual Substitutions

Description	Syntax
FcCosCap	BITS { classF(0), class1(1), class2(2), class3(3), class4(4), class5(5), class6(6) }
FcFeModuleCapacity	Unsigned32
FcFeFxPortCapacity	Unsigned32
FcFeModuleIndex	Unsigned32
FcFeFxPortIndex	Unsigned32
FcFeNxPortIndex	Integer32 (1..126)
FcBbCreditModel	INTEGER { regular(1), alternate (2) }

## Configuration Group

This group consists of scalar objects and tables. It contains the configuration and service parameters of the Fabric Element and the FxPorts. The group represents a set of parameters associated with the Fabric Element or an FxPort to support its NxPorts. The objects described in this section are not in a table format. An example of how to access one of these objects is: "snmpget localhost public fcFeFabricName.0".

### fcFeFabricName

The Name\_Identifier of the Fabric to which this Fabric Element belongs.

#### Syntax

FcNameId

#### Access

read-write

#### Status

Current



### Return Value

The World Wide Name of the switch. For example, 10 00 00 C0 DD 00 71 C9.  
Writes are not supported.

## fcFeElementName

The Name\_Identifier of the Fabric Element.

### Syntax

FcNameId

### Access

read-write

### Status

Current

### Return Value

The World Wide Name of the switch. For example, 10 00 00 C0 DD 00 71 C9.  
Writes are not supported.

## fcFeModuleCapacity

The maximum number of modules in the Fabric Element, regardless of their current state.

### Syntax

FcFeModuleCapacity

### Access

read-only

### Status

Current

### Return Value

The total number of switches in the fabric.

## Module Table

The objects described in this section are in table format indexed by switch. An example of how to access one of these objects is: "snmpget localhost public fcFeModuleDescr.1". This table contains one entry for each module (information of the modules). This table contains one entry for each module, information of the modules.

---

**fcFeModuleDescr**

A textual description of the module. This value should include the full name and version identification of the module.

**Syntax**

SnmpAdminString

**Access**

read-only

**Status**

current

**Return Value**

A configurable description of the module table entry. For example, QLogic SANbox2 FC Switch.

**fcFeModuleObjectID**

The vendor's authoritative identification of the module. This value may be allocated within the SMI enterprises subtree (1.3.6.1.4.1), and provides a means for determining what kind of module is being managed.

For example, this object could take the value 1.3.6.1.4.1.99649.3.9 if vendor "Neufe Inc." was assigned the subtree 1.3.6.1.4.1.99649, and had assigned the identifier 1.3.6.1.4.1.99649.3.9 to its 'FeFiFo-16 PlugInCard'.

**Syntax**

OBJECT IDENTIFIER

**Access**

read-only

**Status**

current

**Return Value**

The module identification number. For example, 1.3.6.1.4.1.1663.1.1.1.1.1.

**fcFeModuleOperStatus**

Switch definitions map 1-to-1 with the MIB definitions. This object indicates the operational status of the module.

- online (1) - the module is functioning properly
- offline (2) - the module is not available

- testing (3) - the module is under testing
- faulty (4) - the module is defective in some way

### Syntax

```
INTEGER {
    online(1), - functional
    offline(2), - not available
    testing(3), - under testing
    faulty(4) - defective
}
```

### Access

read-only

### Status

Current

### Return Value

The operational status of that module.

## fcFeModuleLastChange

This object contains the value of sysUpTime when the module entered its current operational status. A value of zero indicates that the operational status of the module has not changed since the agent last restarted.

### Syntax

TimeStamp

### Access

read-only

### Status

Current

### Return Value

Unsupported. Always returns error status "NoSuchName".

## fcFeModuleFxPortCapacity

The number of FxPort that can be contained within the module. Within each module, the ports are uniquely numbered in the range from 1 to fcFeModuleFxPortCapacity inclusive. However, the numbers are not required to be contiguous.

## FxPort Configuration Table

---

### Syntax

FcFeFxPortCapacity

### Access

read-only

### Status

current

### Return Value

The total number of physical ports on the switch. For example, 16.

## fcFeModuleName

The Name\_Identifier of the switch.

### Syntax

FcNameId

### Access

read-write

### Status

current

### Return Value

The World Wide Name of the switch. Writes are not supported. For example, 10 00 00 C0 DD 00 71 C9.

## FxPort Configuration Table

The objects described in this section are in table format indexed by switch index and port index. An example of how to access one of these objects is: "snmpget localhost public fcFxPortName.1.1. This table contains one entry for each FxPort and Configuration parameters of the ports. This table contains, one entry for each FxPort, configuration parameters of the ports.

## fcFxPortName

The World\_wide Name of this FxPort. Each FxPort has a unique Port World\_wide Name within the Fabric.

### Syntax

FcNameId

### Access

read-only

**Status**

current

**Return Value**

Port World Wide Name. For example, the return value for port #2 would be 20 02 00 C0 DD 00 71 C9, and the return value for port #14 would be 20 0E 00 C0 DD 00 71 C9.

**fcFxPortFcphVersionHigh**

The highest or most recent version of FC-PH that the FxPort is configured to support.

**Syntax**

FcphVersion

**Access**

read-only

**Status**

Current

**Return Value**

Always returns 32 (0x20).

**fcFxPortFcphVersionLow**

The lowest or earliest version of FC-PH that the FxPort is configured to support.

**Syntax**

FcphVersion

**Access**

read-only

**Status**

current

**Return Value**

Always returns 9.

**fcFxPortBbCredit**

The total number of receive buffers available for holding Class 1 connect-request, Class 2 or 3 frames from the attached NxPort. It is for buffer-to-buffer flow control in the direction from the attached NxPort (if applicable) to FxPort.

## FxPort Configuration Table

---

### Syntax

FcBbCredit

### Access

read-only

### Status

current

### Return Value

The default is: 12 (0xC), unless extended credits are used.

## fcFxPortRxBufSize

The largest Data\_Field Size (in octets) for an FT\_1 frame that can be received by the FxPort.

### Syntax

FcRxDataFieldSize

### Access

read-only

### Status

current

### Return Value

Always returns 2112 (0x840).

## fcFxPortRatov

The Resource\_Allocation\_Timeout Value configured for the FxPort. This is used as the timeout value for determining when to reuse an NxPort resource such as a Recovery\_Qualifier. It represents E\_D\_TOV plus twice the maximum time that a frame may be delayed within the fabric and still be delivered. Refer to [“fcFxPort-Edtov” on page 5-9](#) for more information.

### Syntax

MilliSeconds

### Access

read-only

### Status

Current

**Return Value**

The default is: 10000 (0x2710).

**fcFxPortEdtov**

The E\_D\_TOV value configured for the FxPort. The Error\_Detect\_Timeout Value is used as the timeout value for detecting an error condition.

**Syntax**

Milliseconds

**Access**

read-only

**Status**

current

**Return Value**

The default is: 2000 (0x7D0).

**fcFxPortCosSupported**

A value indicating the set of classes of service supported by the FxPort.

**Syntax**

FcCosCap

**Access**

read-only

**Status**

Current

**Return Value**

Always returns Class F, 2, and 3 (0x0d).

**fcFxPortIntermixSupported**

A flag indicating whether or not the FxPort supports an Intermixed Dedicated Connection.

**Syntax**

TruthValue

**Access**

read-only

## Status

current

## Return Value

Always returns False (2).

## fcFxPortStackedConnMode

A value indicating the mode of Stacked Connect supported by the FxPort.

## Syntax

FcStackedConnMode

## Access

read-only

## Status

current

## Return Value

Always returns None (1).

## fcFxPortClass2SeqDeliv

A flag indicating whether or not Class 2 Sequential Delivery is supported by the FxPort.

## Syntax

TruthValue

## Access

read-only

## Status

current

## Return Value

Always returns True (1).

## fcFxPortClass3SeqDeliv

A flag indicating whether or not Class 3 Sequential Delivery is supported by the FxPort.

## Syntax

TruthValue



**Access**

read-only

**Status**

current

**Return Value**

Always returns True (1).

**fcFxPortHoldTime**

The maximum time, in microseconds, that the FxPort shall hold a frame before discarding the frame if it is unable to deliver the frame. The value 0 means that the FxPort does not support this parameter.

**Syntax**

MicroSeconds

**Access**

read-only

**Status**

current

**Return Value**

The default ED\_TOV parameter is: 2000 (0x7D0).

**The Status Group**

The objects described in this section are in table format indexed by switch index and port index. An example of how to access one of these objects is: "snmpget localhost public fcFxPortId.1.1". This group consists of tables that contain operational status and established service parameters for the Fabric Element and the attached NxPorts.

This group consists of tables that contains operational status and established service parameters for the Fabric Element and the attached NxPorts. This table contains, one entry for each FxPort, the operational status and parameters of the FxPorts.

**fcFxPortID**

The address identifier by which this FxPort is identified within the fabric. The FxPort may assign its address identifier to its attached NxPort(s) during Fabric Login.

**Syntax**

FcAddressId

**Access**

read-only

**Status**

current

**Return Value**

The address for each port based on Domain, Area, and ALPA.

**fcFxPortBbCreditAvailable**

The number of buffers currently available for receiving frames from the attached port in the buffer-to-buffer flow control. The value should be less than or equal to fcFxPortBbCredit.

**Syntax**

Gauge32

**Access**

read-only

**Status**

Current

**Return Value**

Unsupported. Always returns error status "NoSuchName".

**fcFxPortOperMode**

The current operational mode of the FxPort.

**Syntax**

INTEGER { unknown(1), fPort(2), flPort(3) }

**Access**

read-only

**Status**

current

**Return Value**Refer to [Table 5-2](#) for fcFxPortOperMode return values.

**Table 5-2 Port Operational Modes**

Mode	Return Value
Unknown	1
F_Port	2
FL_Port	3

## fcFxPortAdminMode

The desired operational mode of the FxPort.

### Syntax

INTEGER { fPort(2), flPort(3) }

### Access

read-write

### Status

Current

### Return Value

Refer to [Table 5-3](#) for fcFxPortAdminMode return values.

**Table 5-3 Port Administrative Modes**

Mode	Value
F_Port	2
FL_Port	3

## FxPort Physical Level Table

The objects described in this section are in table format indexed by switch index and port index. An example of how to access one of these objects is: "snmpget localhost public fcFxPortPhysAdminStatus.1.1". This table contains one entry for each FxPort in the Fabric Element, the physical level status, and parameters of the FxPorts.

This table contains, one entry for each FxPort in the Fabric Element, the physical level status and parameters of the FxPorts.

## fcFxPortPhysAdminStatus

The desired state of the FxPort. A management station may place the FxPort in a desired state by setting this object accordingly. The testing(3) state indicates that no operational frames can be passed. When a Fabric Element initializes, all FxPorts start with fcFxPortPhysAdminStatus in the offline(2) state. As the result of either explicit management action or per configuration information accessible by the Fabric Element, fcFxPortPhysAdminStatus is then changed to either the online(1) or testing(3) states, or remains in the offline state.

### Syntax

```
INTEGER {
    online(1), - place port online
    offline(2), - take port offline
    testing(3) - initiate test procedures
}
```

### Access

read-write

### Status

current

### Return Value

Refer to [Table 5-4](#) for fcFxPortPhysAdminStatus read values.

**Table 5-4 fcFxPortPhysAdminStatus Read Return Values**

Port	Value
Online	online (1)
Offline	offline (2)
Diagnostic	testing (3)
Down	offline (2)

Refer to [Table 5-5](#) for fcFxPortPhysAdminStatus write values.

**Table 5-5 fcFxPortPhysAdminStatus Write Values**

Port	Value
Online	online (1)

**Table 5-5 fcFxPortPhysAdminStatus Write Values**

Port	Value
Offline	offline (2)
Diagnostic	testing (3)

## fcFxPortPhysOperStatus

The current operational status of the FxPort. The testing(3) indicates that no operational frames can be passed. If fcFxPortPhysAdminStatus is offline(2), then fcFxPortPhysOperStatus should be offline(2). If fcFxPortPhysAdminStatus is changed to online(1), then fcFxPortPhysOperStatus should change to online(1). If the FxPort is ready to accept Fabric Login request from the attached NxPort, it should proceed and remain in the link- failure(4) state if, and only if, there is a fault that prevents it from going to the online(1) state.

### Syntax

```

INTEGER {
    online(1), - Login may proceed
    offline(2), - Login cannot proceed
    testing(3), - port is under test
    linkFailure(4) - failure after online/testing
}

```

### Access

read-only

### Status

current

### Return Value

Refer to [Table 5-6](#) for fcFxPortPhysOperStatus return values.

**Table 5-6 fcFxPortPhysOperStatus Return Values**

Status	Return Value
Online	online (1)
Offline	offline (2)
Diagnostic	testing (3)

**Table 5-6 fcFxPortPHysOperStatus Return Values**

Status	Return Value
Down	linkfailure (4)

**fcFxPortPhysLastChange**

The value of sysUpTime at the time the FxPort entered its current operational status. A value of zero indicates that the FxPort's operational status has not changed since the agent last restarted.

**Syntax**

TimeStamp

**Access**

read-only

**Status**

current

**Return Value**

Unsupported. Always returns error status "NoSuchName".

**fcFxPortPhysRttov**

The Receiver\_Transmitter\_Timeout value of the FxPort. This is used by the receiver logic to detect a loss of synchronization.

**Syntax**

MilliSeconds

**Access**

read-write

**Status**

current

**Return Value**

The default RT\_TOV parameter is: 100 (0x64). This is a global setting for the switch. If writing value to a port, all ports will reflect this new value.

**Fx Port Fabric Login Table**

The objects described in this section are in table format indexed by switch index and port index. An example of how to access one of these objects is: "snmpget localhost public fcFxPortFcphVersionAgreed.1.1". This table contains one entry

for each FxPort in the Fabric Element and the Service Parameters that have been established from the most recent Fabric Login (implicit or explicit).

This table contains, one entry for each FxPort in the Fabric Element, the Service Parameters that have been established from the most recent Fabric Login, implicit or explicit.

### **fcFxPortFcphVersionAgreed**

The version of FC-PH that the FxPort has agreed to support from the Fabric Login.

#### **Syntax**

FcphVersion

#### **Access**

read-only

#### **Status**

current

#### **Return Value**

Unsupported.

### **fcFxPortNxPortBbCredit**

The total number of buffers available for holding Class 1 connect-request, Class 2, or Class 3 frames to be transmitted to the attached NxPort. It is for buffer-to-buffer flow control in the direction from FxPort to NxPort. The buffer-to-buffer flow control mechanism is indicated in the respective fcFxPortBbCreditModel.

#### **Syntax**

FcBbCredit

#### **Access**

read-only

#### **Status**

current

#### **Return Value**

Unsupported.

### **fcFxPortNxPortRxDataFieldSize**

The Receive Data Field Size of the attached NxPort. This object specifies the largest Data Field Size for an FT\_1 frame that can be received by the NxPort.

## Fx Port Fabric Login Table

---

### Syntax

FcRxDataFieldSize

### Access

read-only

### Status

current

### Return Value

Unsupported.

## fcFxPortCosSuppAgreed

A variable indicating that the attached NxPort has requested the FxPort for the support of classes of services and the FxPort has granted the request.

### Syntax

FcCosCap

### Access

read-only

### Status

current

### Return Value

Unsupported.

## fcFxPortIntermixSuppAgreed

A variable indicating that the attached NxPort has requested the FxPort for the support of Intermix and the FxPort has granted the request. This flag is only valid if Class 1 service is supported.

### Syntax

TruthValue

### Access

read-only

### Status

current

### Return Value

Unsupported.



**fcFxPortStackedConnModeAgreed**

A variable indicating whether the FxPort has agreed to support stacked connect from the Fabric Login. This is only meaningful if the ports are using Class 1 service.

**Syntax**

FcStackedConnMode

**Access**

read-only

**Status**

current

**Return Value**

Unsupported.

**fcFxPortClass2SeqDelivAgreed**

A variable indicating whether the FxPort has agreed to support Class 2 sequential delivery from the Fabric Login. This is only meaningful if the ports are using Class 2 service.

**Syntax**

TruthValue

**Access**

read-only

**Status**

Current

**Return Value**

Unsupported.

**fcFxPortClass3SeqDelivAgreed**

A flag indicating whether the FxPort has agreed to support Class 3 sequential delivery from the Fabric Login. This is only meaningful if the ports are using Class 3 service.

**Syntax**

TruthValue

**Access**

read-only

**Status**

current

**Return Value**

Unsupported.

**fcFxPortNxPortName**

The port name of the attached NxPort.

**Syntax**

FcNameId

**Access**

read-only

**Status**

Current

**Return Value**

Unsupported.

**fcFxPortConnectedNxPort**

The address identifier of the destination NxPort with which this FxPort is currently engaged in a either a Class 1 or loop connection. If this FxPort is not engaged in a connection, then the value of this object is “000000”H.

**Syntax**

FcAddressId

**Access**

read-only

**Status**

Current

**Return Value**

Unsupported.

**fcFxPortBbCreditModel**

This object identifies the BB\_Credit model used by the FxPort.

**Syntax**

FcBbCreditModel

**Access**

read-write

**Status**

current

**Return Value**

Unsupported.

## The Error Group

The objects described in this section are in table format indexed by switch index and port index. An example of how to access one of these objects is: "snmpget localhost public fcFxPortLinkFailures.1.1". This group consists of tables that contain information about the various types of errors detected. The management station may use the information in this group to determine the quality of the link between the FxPort and its attached NxPort.

The FxPort Error table contains, one entry for each FxPort in the Fabric Element, counters recording numbers of errors detected since the management agent reinitialized. The first 6 columnar objects after the port index corresponds to the counters in the Link Error Status Block.

### fcFxPortLinkFailures

The number of link failures detected by this FxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of LinkFailures encountered for a port.

### fcFxPortSyncLosses

The number of loss of synchronization's detected by the FxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of LossOfSync's encountered for a port.

**fcFxpPortSigLosses**

The number of loss of signals detected by the FxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

Unsupported. Always returns error status "NoSuchName".

**fcFxpPortPrimSeqProtoErrors**

The number of primitive sequence protocol errors detected by the FxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of PrimSeqErrors encountered for a port.

**fcFxpPortInvalidTxWords**

The number of invalid transmission words detected by the FxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of DecodeError's encountered for a port.

**fcFxpPortInvalidCrcs**

The number of invalid CRC's detected by this FxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of InvalidCRC's encountered for a port.

**fcFxpPortDelimiterErrors**

The number of Delimiter Errors detected by this FxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

Unsupported. Always returns error status "NoSuchName".

**fcFxpPortAddressIdErrors**

The number of address identifier errors detected by this FxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of InvDestAddr's encountered for a port.

**fcFxPortLinkResetIns**

The number of Link Reset Protocol's received by this FxPort from the attached NxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of RxLinkResets received by a port.

**fcFxPortLinkResetOuts**

The number of Link Reset Protocol's issued by this FxPort to the attached NxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of TxLinkResets sent by a port.

**fcFxPortOlsIns**

The number of Offline Sequences received by this FxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of RxOfflineSeq's received by a port.

**fcFxPortOlsOuts**

The number of Offline Sequences issued by this FxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of TxOfflineSeq's sent by a port.

**Accounting Groups**

Each group consists of a table that contains accounting information for the FxPorts in the Fabric Element: Class 1 Accounting Group, Class 2 Accounting Group, and Class 3 Accounting Group.

**Class 1 Accounting Table**

The objects described in this section are in table format indexed by switch index and port index. An example of how to access one of these objects is: "snmpget localhost public fcFxPortC1InFrames.1.1". This table contains one entry for each FxPort in the Fabric Element and Counter32s for certain types of events occurred in the FxPorts since the management agent was re-initialized.

**fcFxPortC1InFrames**

The number of Class 1 frames (other than Class 1 connect-request) received by this FxPort from its attached NxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

Class 1 currently not supported. Always returns error status "NoSuchName".

**fcFxPortC1OutFrames**

The number of Class 1 frames (other than Class 1 connect- request) delivered through this FxPort to its attached NxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

Class 1 currently not supported. Always returns error status "NoSuchName".

**fcFxPortC1InOctets**

The number of Class 1 frame octets, including the frame delimiters, received by this FxPort from its attached NxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

Unsupported. Always returns error status "NoSuchName".



**fcFxpPortC1OutOctets**

The number of Class 1 frame octets, including the frame delimiters, delivered through this FxPort its attached NxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

Unsupported. Always returns error status "NoSuchName".

**fcFxpPortC1Discards**

The number of Class 1 frames discarded by this FxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

Unsupported. Always returns error status "NoSuchName".

**fcFxpPortC1FbsyFrames**

The number of F\_BSY frames generated by this FxPort against Class 1 connect-request.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

Unsupported. Always returns error status "NoSuchName".

**fcFxPortC1FrjtFrames**

The number of F\_RJT frames generated by this FxPort against Class 1 connect-request.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

Unsupported. Always returns error status "NoSuchName".

**fcFxPortC1InConnections**

The number of Class 1 connections successfully established in which the attached NxPort is the source of the connect-request.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

Unsupported. Always returns error status "NoSuchName".

**fcFxPortC1OutConnections**

The number of Class 1 connections successfully established in which the attached NxPort is the destination of the connect-request.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

Unsupported. Always returns error status "NoSuchName".

**fcFxPortC1ConnTime**

The cumulative time that this FxPort has been engaged in Class 1 connection. The amount of time is counted from after a connect-request has been accepted until the connection is disengaged, either by an EOFdt or Link Reset.

**Syntax**

Milliseconds

**Access**

read-only

**Status**

current

**Return Value**

Unsupported. Always returns error status "NoSuchName".

**Class 2 Accounting Table**

The objects described in this section are in table format indexed by switch index and port index. An example of how to access one of these objects is: "snmpget localhost public fcFxPortC2InFrames.1.1". This table contains one entry for each FxPort in the Fabric Element and Counter32s for certain types of events occurred in the FxPorts since the management agent was re-initialized.

**fcFxPortC2InFrames**

The number of Class 2 frames received by this FxPort from its attached NxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of Class2FramesIn received by a port.

### **fcFxPortC2OutFrames**

The number of Class 2 frames delivered through this FxPort to its attached NxPort.

#### **Syntax**

Counter32

#### **Access**

read-only

#### **Status**

current

#### **Return Value**

The total number of Class2FramesOut sent by a port.

### **fcFxPortC2InOctets**

The number of Class 2 frame octets, including the frame delimiters, received by this FxPort from its attached NxPort.

#### **Syntax**

Counter32

#### **Access**

read-only

#### **Status**

current

#### **Return Value**

The total number of Class2WordsIn received by a port.

### **fcFxPortC2OutOctets**

The number of Class 2 frame octets, including the frame delimiters, delivered through this FxPort to its attached NxPort.

#### **Syntax**

Counter32

#### **Access**

read-only

#### **Status**

current

**Return Value**

The total number of Class2WordsOut sent by a port.

**fcFxPortC2Discards**

The number of Class 2 frames discarded by this FxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of Class2Toss discarded by a port.

**fcFxPortC2FbsyFrames**

The number of F\_BSY frames generated by this FxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of FBusy frames generated by this port for Class 2 and 3 frames.

**fcFxPortC2FrjtFrames**

The number of F\_RJT frames generated by this FxPort against Class 2 frames.

**Syntax**

Counter32

**Access**

read-only

## Class 3 Accounting Table

---

### Status

current

### Return Value

The total number of FReject frames generated by this port for Class 2 and 3 frames.

## Class 3 Accounting Table

The objects described in this section are in table format indexed by switch index and port index. An example of how to access one of these objects is: "snmpget localhost public fcFxPortC3InFrames.1.1". This table contains one entry for each FxPort in the Fabric Element and Counter32s for certain types of events occurred in the FxPorts since the management agent has re-initialized.

### fcFxPortC3InFrames

The number of Class 3 frames received by this FxPort from its attached NxPort.

#### Syntax

Counter32

#### Access

read-only

#### Status

current

#### Return Value

The total number of Class3FramesIn received by a port.

### fcFxPortC3OutFrames

The number of Class 3 frames delivered through this FxPort to its attached NxPort.

#### Syntax

Counter32

#### Access

read-only

#### Status

current

#### Return Value

The total number of Class3FramesOut sent by a port.

**fcFxPortC3InOctets**

The number of Class 3 frame octets, including the frame delimiters, received by this FxPort from its attached NxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of Class3WordsOut received by a port.

**fcFxPortC3OutOctets**

The number of Class 3 frame octets, including the frame delimiters, delivered through this FxPort to its attached NxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

**Return Value**

The total number of Class3WordsOut sent by a port.

**fcFxPortC3Discards**

The number of Class 3 frames discarded by this FxPort.

**Syntax**

Counter32

**Access**

read-only

**Status**

current

### Return Value

The total number of Class3Toss discarded by a port.

## Capability Group

The objects described in this section are in table format indexed by switch index and port index. An example of how to access one of these objects is: "snmpget localhost public fcFxPortName.1.1". The Capability Group consists of a table describing information about what each FxPort is inherently capable of operating or supporting. A capability may be used as expressed in its respective object value in the Configuration group.

### fcFxPortCapFcphVersionHigh

The highest or most recent version of FC-PH that the FxPort is capable of supporting.

#### Syntax

FcphVersion

#### Access

read-only

#### Status

current

#### Return Value

Always returns 32 (0x20).

### fcFxPortCapFcphVersionLow

The lowest or earliest version of FC-PH that the FxPort is capable of supporting.

#### Syntax

FcphVersion

#### Access

read-only

#### Status

current

#### Return Value

Always returns 9.



---

**fcFxPortCapBbCreditMax**

The maximum number of receive buffers available for holding Class 1 connect-request, Class 2, or Class 3 frames from the attached NxPort.

**Syntax**

FcBbCredit

**Access**

read-only

**Status**

current

**Return Value**

The default is: 255 (0xFF).

**fcFxPortCapBbCreditMin**

The minimum number of receive buffers available for holding Class 1 connect-request, Class 2, or Class 3 frames from the attached NxPort.

**Syntax**

FcBbCredit

**Access**

read-only

**Status**

current

**Return Value**

The default is: 0 (0x00).

**fcFxPortCapRxDataFieldSizeMax**

The maximum size in bytes of the Data Field in a frame that the FxPort is capable of receiving from its attached NxPort.

**Syntax**

FcRxDataFieldSize

**Access**

read-only

**Status**

current

**Return Value**

2112 (0x840).

**fcFxPortCapRxDataFieldSizeMin**

The minimum size in bytes of the Data Field in a frame that the FxPort is capable of receiving from its attached NxPort.

**Syntax**

FcRxDataFieldSize

**Access**

read-only

**Status**

current

**Return Value**

128 (0x80).

**fcFxPortCapCos**

A value indicating the set of classes of service that the FxPort is capable of supporting.

**Syntax**

FcCosCap

**Access**

read-only

**Status**

current

**Return Value**

Always returns Class F, 2, and 3 (0x0d).

**fcFxPortCapIntermix**

A flag indicating whether or not the FxPort is capable of supporting the inter-mixing of Class 2 and Class 3 frames during a Class 1 connection. This flag is only valid if the port is capable of supporting Class 1 service.

**Syntax**

TruthValue

**Access**

read-only

**Status**

current

**Return Value**

Always returns False (2).

**fcFxPortCapStackedConnMode**

A value indicating the mode of Stacked Connect request that the FxPort is capable of supporting.

**Syntax**

FcStackedConnMode

**Access**

read-only

**Status**

current

**Return Value**

Always returns None (1).

**fcFxPortCapClass2SeqDeliv**

A flag indicating whether or not the FxPort is capable of supporting Class 2 Sequential Delivery.

**Syntax**

TruthValue

**Access**

read-only

**Status**

current

**Return Value**

Always returns Yes (1).

**fcFxPortCapClass3SeqDeliv**

A flag indicating whether or not the FxPort is capable of supporting Class 3 Sequential Delivery.

**Syntax**

TruthValue

**Access**

read-only

**Status**

current

**Return Value**

Always returns Yes (1).

**fcFxPortCapHoldTimeMax**

The maximum holding time that the FxPort is capable of supporting, in microseconds.

**Syntax**

MicroSeconds

**Access**

read-only

**Status**

current

**Return Value**

20000 (0x4E20)

**fcFxPortCapHoldTimeMin**

The minimum holding time that the FxPort is capable of supporting, in microseconds.

**Syntax**

MicroSeconds

**Access**

read-only

**Status**

current

**Return Value**

10 (0x0A)

# Glossary

---

## **AL\_PA**

Arbitrated Loop Physical Address

## **Arbitrated Loop**

A Fibre Channel topology where ports use arbitration to establish a point-to-point circuit.

## **Arbitrated Loop Physical Address (AL\_PA)**

A unique one-byte valid value assigned during loop initialization to each NL\_Port on a Loop.

## **Abstract Syntax Notation (ASN.1)**

Abstract Syntax Notation number One (ASN.1) is an international standard that aims at specifying data used in communication protocols.

## **Authentication Trap**

Enables or disables the reporting of SNMP authentication failures. If enabled, a notification trap is sent to the configured trap addresses in the event of an authentication failure if incorrect community string values are used. The default value is False.

## **BER**

Bit Error Rate

## **Bit Error Rate**

The probability that a transmitted bit will be erroneously received. The BER is measured by counting the number of bits in error at the output of a receiver and dividing by the total number of bits in the transmission. BER is typically expressed as a negative power of 10.

## **Buffer Credit**

A measure of port buffer capacity equal to one frame.

## **Class 2 Service**

A service which multiplexes frames at frame boundaries to or from one or more N\_Ports with acknowledgment provided.

## **Class 3 Service**

A service which multiplexes frames at frame boundaries to or from one or more N\_Ports without acknowledgment.

## **Contact**

Contact Specifies the name of the contact person who is to be contacted to respond to trap events. The default is undefined.

## **Datagram**

A message sent between two communicating entities for which no explicit link level acknowledgment is expected.

## **Domain ID**

User defined name that identifies the switch in the fabric.

## **Fabric Management Switch**

The switch through which the fabric is managed.

## **Flash Memory**

Memory on the switch that contains the chassis control firmware.

## **Frame**

Data unit consisting of a start-of-frame (SOF) delimiter, header, data payload, CRC, and an end-of-frame (EOF) delimiter.

## **ICMP**

Internet Control Message Protocol

## **IETF**

Internet Engineering Task Force

## Initiator

The device that initiates a data exchange with a target device.

## Internet Engineering Task Force

A large open international community of network designers, operators, vendors, and researchers concerned with evolution and smooth operation of the Internet, and responsible for producing RFC's. The standards body responsible for Internet standards, including SNMP, TCP/IP and policy for QoS.

## Internet Control Message Protocol

A control protocol strongly related to IP and TCP, and used to convey a variety of control and error indications.

## InteropCredit

This variable determines the number of credits we will advertise on an ISL. Older versions of Brocade software required that we match their offering (decimal value, 0-255).

## IP

Internet Protocol

## ISLSecurity

ISLSecurity determines which switches a port will establish a link with. ANY - we will link with any switch. Ours - we will only link to another QLogic switch. None - the port will not establish an ISL link.

## LCFEnable

LCFEnable gives preference to Link control frames (such as Class 2 ACK frames) over other frames, when queued for transmission in the switch. This may provide better performance when running Class 2 traffic. LCFEnable is incompatible with MFSEnable, and both cannot be selected. (True / False)

## LIP

Loop Initialization Primitive sequence

## Location

Specifies the switch location. The default is undefined.

## Authentication

## Logged-In LED

A port LED that indicates device login or loop initialization status.

## Management Information Base

A set of guidelines and definitions for the Fibre Channel functions. The specification and formal description of a set of objects and variables that can be read and possibly written using the SNMP protocol. Various standard MIBs are defined by the Internet Engineering Task Force.

## Management Workstation

PC workstation that manages the fabric through the fabric management switch.

## MIB

Management Information Base

## MSEnable

Determines whether GS-3 management server commands will be accepted on the port. It can be used to prevent in-band management of the switch on any or all ports. (True / False)

## NL\_Port

Node Loop Port. A Fibre Channel device port that supports arbitrated loop protocol.

## N\_Port

Node Port. A Fibre Channel device port in a point-to-point or fabric connection.

## NMS

Network Management Station

## Network Management Station

The console through which an administrator performs management functions.

## NoClose

Causes the switch to keep the loop open, if no other device is arbitrating. It is intended to improve performance when there is a single L\_Port device connected to the switch. (True / False).

**Node**

An addressable entity connected to an I/O bus or network. Used primarily to refer to computers, storage devices, and storage subsystems. The component of a node that connects to the bus or network is a port.

**Object**

In the context of access control, an entity to which access is controlled and/or usage of which is restricted to authorized subjects.

**QoS**

Quality of Service

**POST**

Power On Self Test

**Power On Self Test (POST)**

Diagnostics that the switch chassis performs at start up.

**Private Device**

A device that can communicate only with other devices on the same loop.

**Private Loop**

A loop of private devices connected to a single switch port.

**ReadCommunity**

Read Community Authentication. A write-only field; the value on the switch and the SNMP management server must be the same. The default value is Public.

**Request For Comment (RFC)**

Internet -related specifications, including standards, experimental definitions, informational documents and best practice definitions, produced by the IETF.

**SANbox Manager**

Switch management application.

**SFF**

Small Form-Factor transceiver.

**SFP**

Small Form-Factor Pluggable. A transceiver device, smaller than a GigaBit Interface Converter, that plugs into the Fibre Channel port.

**Simple Network Management Protocol**

The protocol governing network management and that allows monitoring of network devices.

**SMI**

Structure of Management Information

**Small Form Factor**

A transceiver device, smaller than a GigaBit Interface Converter, that is permanently attached to the circuit board.

**Small Form-Factor Pluggable**

A transceiver device, smaller than a GigaBit Interface Converter, that plugs into the Fibre Channel port.

**SNMP**

Simple Network Management Protocol

**Structure of Management Information**

A notation for setting or retrieving management variables over SNMP.

**Target**

A storage device that responds to an initiator device.

**TCP**

Transmission Control Protocol

**Trap Address**

Specifies the IP address to which SNMP traps are sent. The default is 127.0.0.1. A maximum of 5 trap addresses are supported.

**Trap Community**

Trap Community Authentication. A write-only field; the value on the switch and the SNMP management server must be the same. The default value is Public.

**Trap Enabled**

Check box to enable or disable the trap.

---

**Trap Port**

The port number on which the trap is set.

**Trap Severity**

Specifies a severity level to assign to the trap. Open the pull-down menu and choose a level. Traps must be enabled to access this pull-down menu. Trap severity levels include Unknown, Emergency, Alert, Critical, Error, Warning, Notify, Info, Debug, and Mark

**UDP**

User Datagram Protocol

**User Datagram Protocol**

An Internet protocol that provides connectionless datagram delivery service to applications. Abbreviated UDP. UDP over Internet Protocol adds the ability to address multiple endpoints within a single network node to IP.

**VIEnable**

FC-VI. When enabled, VI preference frames will be transmitted ahead of other frames. (True / False)

**Worldwide Name (WWN)**

A unique 64-bit address assigned to a device by the device manufacturer.

**Write Community**

Write Community Authentication. A write-only field; the value on the switch and the SNMP management server must be the same. The default value is Private.

**WWN**

Worldwide Name

**Zone**

A set of ports or devices grouped together to control the exchange of information.

**Zone Set**

A set of zones grouped together. The active zone set defines the zoning for a fabric.



## A

- Accounting Groups 5-25
- Additional IP Objects 3-27
- Additional TCP Objects 3-40
- Address Translation Group 3-12
- Agent 1-1
- Alert 1-3, 4-76
- atIfIndex 3-12
- atNetAddress 3-13
- atPhysAddress 3-12
- atTable 3-12

## C

- Capability Group 5-34
- Class 1 Accounting Table 5-25
- Class 2 Accounting Table 5-29
- Class 3 Accounting Table 5-32
- Configuration Group 5-2
- configurationChangeTime 4-4
- configuring a switch using SANbox Manager 2-3
- Connectivity Table 4-5
- Connectivity Unit Group 4-3
- connUnitConfigurationChangeTime 4-12
- connUnitContact 4-15
- connUnitControl 4-14
- connUnitDomainId 4-9
- connUnitEventCurrId 4-18
- connUnitEventDescr 4-44
- connUnitEventFilter 4-16
- connUnitEventId 4-41
- connUnitEventIndex 4-40
- connUnitEventObject 4-43
- connUnitEventSeverity 4-42
- connUnitEventType 4-43
- connUnitEventUnitId 4-40
- connUnitGlobalId 4-6
- connUnitId 4-5
- connUnitInfo 4-13
- connUnitLinkAgentAddressTypeY 4-48
- connUnitLinkAgentAddressY 4-48
- connUnitLinkAgentPortY 4-49
- connUnitLinkConnIdY 4-49
- connUnitLinkCurrIndex 4-50
- connUnitLinkIndex 4-45
- connUnitLinkNodeIdx 4-45
- connUnitLinkNodeIdY 4-47
- connUnitLinkPortNumber 4-47
- connUnitLinkPortNumberX 4-46
- connUnitLinkPortWwnX 4-46
- connUnitLinkPortWwnY 4-47
- connUnitLinkUnitId 4-45
- connUnitLinkUnitTypeY 4-49
- connUnitLocation 4-16
- connUnitMaxEvents 4-17
- connUnitModuleId 4-13
- connUnitName 4-13
- connUnitNumEvents 4-17
- connUnitNumports 4-6
- connUnitNumRevs 4-12
- connUnitNumSensors 4-11
- connUnitNumZones 4-12
- connUnitPortControl 4-34
- connUnitPortFCClassCap 4-28
- connUnitPortFCClassOp 4-29
- connUnitPortFCId 4-32
- connUnitPortHWState 4-39
- connUnitPortIndex 4-26
- connUnitPortModuleType 4-31
- connUnitPortName 4-36
- connUnitPortNodeWwn 4-39
- connUnitPortPhysicalNumber 4-37
- connUnitPortProtocolCap 4-38
- connUnitPortProtocolOp 4-38
- connUnitPortRevision 4-33
- connUnitPortSn 4-33
- connUnitPortSpeed 4-34
- connUnitPortStatCountAddressErrors 4-68
- connUnitPortStatCountBBCreditZero 4-53
- connUnitPortStatCountClass1FBSYFrames 4-56
- connUnitPortStatCountClass1FRJTFrames 4-57
- connUnitPortStatCountClass1PBSYFrames 4-57
- connUnitPortStatCountClass1PRJTFrames 4-58
- connUnitPortStatCountClass1RxFrames 4-56
- connUnitPortStatCountClass1TxFrames 4-56
- connUnitPortStatCountClass2FBSYFrames 4-59
- connUnitPortStatCountClass2FRJTFrames 4-59
- connUnitPortStatCountClass2PBSYFrames 4-59
- connUnitPortStatCountClass2PRJTFrames 4-60
- connUnitPortStatCountClass2RxFrames 4-58
- connUnitPortStatCountClass2TxFrames 4-58
- connUnitPortStatCountClass3Discards 4-61
- connUnitPortStatCountClass3RxFrames 4-60
- connUnitPortStatCountClass3TxFrames 4-61
- connUnitPortStatCountDelimiterErrors 4-69
- connUnitPortStatCountEncodingDisparityErrors 4-69

connUnitPortStatCountError 4-51  
 connUnitPortStatCountFBSYFrames 4-54  
 connUnitPortStatCountFramesTooLong 4-68  
 connUnitPortStatCountFramesTruncated 4-68  
 connUnitPortStatCountFRJTFrames 4-55  
 connUnitPortStatCountInputBuffersFull 4-54  
 connUnitPortStatCountInvalidCRC 4-65  
 connUnitPortStatCountInvalidOrderedSets 4-67  
 connUnitPortStatCountInvalidTxWords 4-66  
 connUnitPortStatCountLinkFailures 4-65  
 connUnitPortStatCountLossofSignal 4-66  
 connUnitPortStatCountLossofSynchronization 4-67  
 connUnitPortStatCountNumberLinkResets 4-63  
 connUnitPortStatCountNumberOfflineSequences 4-65  
 connUnitPortStatCountPBSYFrames 4-54  
 connUnitPortStatCountPrimitiveSequenceProtocolErrors 4-66  
 connUnitPortStatCountPRJTFrames 4-55  
 connUnitPortStatCountRxBroadcastObjects 4-62  
 connUnitPortStatCountRxElements 4-53  
 connUnitPortStatCountRxLinkResets 4-63  
 connUnitPortStatCountRxMulticastObjects 4-61  
 connUnitPortStatCountRxObjects 4-52  
 connUnitPortStatCountRxOfflineSequences 4-64  
 connUnitPortStatCountTxBroadcastObjects 4-62  
 connUnitPortStatCountTxElements 4-52  
 connUnitPortStatCountTxLinkResets 4-63  
 connUnitPortStatCountTxMulticastObjects 4-62  
 connUnitPortStatCountTxObjects 4-51  
 connUnitPortStatCountTxOfflineSequences 4-64  
 connUnitPortState1 4-29  
 connUnitPortStatIndex 4-51  
 connUnitPortStatObject 4-37  
 connUnitPortStatUnitId 4-50  
 connUnitPortStatus 4-30  
 connUnitPortTransmitterType 4-30  
 connUnitPortType 4-27  
 connUnitPortUnitId 4-26  
 connUnitPortWwn 4-32  
 connUnitPrincipal 4-10  
 connUnitProduct 4-8  
 connUnitProxyMaster 4-10  
 connUnitREventTime 4-42  
 connUnitRevsDescription 4-20  
 connUnitRevsIndex 4-19  
 connUnitRevsRevId 4-19  
 connUnitRevsUnitId 4-18  
 connUnitSensorCharacteristic 4-25  
 connUnitSensorIndex 4-21  
 connUnitSensorInfo 4-23  
 connUnitSensorMessage 4-23  
 connUnitSensorName 4-21  
 connUnitSensorStatus 4-22  
 connUnitSensorType 4-24

connUnitSensorUnitId 4-20  
 connUnitSEventTime 4-42  
 connUnitSn 4-8  
 connUnitSnsClassOfSvc 4-72  
 connUnitSnsFabricPortName 4-74  
 connUnitSnsFC4Type 4-73  
 connUnitSnsHardAddress 4-75  
 connUnitSnsId 4-70  
 connUnitSnsNodeIPAddress 4-72  
 connUnitSnsNodeName 4-71  
 connUnitSnsPortIdentifier 4-71  
 connUnitSnsPortIndex 4-70  
 connUnitSnsPortIPAddress 4-74  
 connUnitSnsPortName 4-71  
 connUnitSnsPortType 4-73  
 connUnitSnsProcAssoc 4-72  
 connUnitSnsSymbolicNodeName 4-75  
 connUnitSnsSymbolicPortName 4-75  
 connUnitState 4-7  
 connUnitStatus 4-7  
 connUnitStatusChangeTime 4-11  
 connUnitTableChangeTime 4-5  
 connUnitType 4-6  
 connUnitUpTime 4-9  
 connUnitUrl 4-9  
 Critical 1-3, 4-76

## D

Debug 1-3, 4-76

## E

EGP Group 3-43  
 EGP Neighbor Table 3-44  
 egpAs 3-48  
 egpInErrors 3-43  
 egpInMsgs 3-43  
 egpNeighAddr 3-44  
 egpNeighAs 3-45  
 egpNeighEventTrigger 3-48  
 egpNeighInErrMsgs 3-46  
 egpNeighInErrs 3-45  
 egpNeighInMsgs 3-45  
 egpNeighIntervalHello 3-47  
 egpNeighIntervalPoll 3-47  
 egpNeighMode 3-48  
 egpNeighOutErrMsgs 3-46  
 egpNeighOutErrs 3-46  
 egpNeighOutMsgs 3-45  
 egpNeighState 3-44  
 egpNeighStateDowns 3-47  
 egpNeighStateUps 3-47  
 egpNeighTable 3-44  
 egpOutErrors 3-44

eggOutMsgs 3-43  
Emergency 1-3, 4-76  
Error 1-3, 4-76  
Error Group 5-21  
Event Table 4-40

## F

fcFeElementName 5-3  
fcFeFabricName 5-2  
fcFeModuleCapacity 5-3  
fcFeModuleDescr 5-4  
fcFeModuleFxpPortCapacity 5-5  
fcFeModuleLastChange 5-5  
fcFeModuleName 5-6  
fcFeModuleObjectID 5-4  
fcFeModuleOperStatus 5-4  
fcFeModuleTable 5-4  
fcFxpPortAddressIdErrors 5-23  
fcFxpPortAdminMode 5-13  
fcFxpPortBbCredit 5-7  
fcFxpPortBbCreditAvailable 5-12  
fcFxpPortBbCreditModel 5-20  
fcFxpPortC1AccountingTable 5-25  
fcFxpPortC1ConnTime 5-29  
fcFxpPortC1Discards 5-27  
fcFxpPortC1FbsyFrames 5-27  
fcFxpPortC1FrjtFrames 5-28  
fcFxpPortC1InConnections 5-28  
fcFxpPortC1InFrames 5-25  
fcFxpPortC1InOctets 5-26  
fcFxpPortC1OutConnections 5-28  
fcFxpPortC1OutFrames 5-26  
fcFxpPortC1OutOctets 5-27  
fcFxpPortC2AccountingTable 5-29  
fcFxpPortC2Discards 5-31  
fcFxpPortC2FbsyFrames 5-31  
fcFxpPortC2FrjtFrames 5-31  
fcFxpPortC2InFrames 5-29  
fcFxpPortC2InOctets 5-30  
fcFxpPortC2OutFrames 5-30  
fcFxpPortC2OutOctets 5-30  
fcFxpPortC3Discards 5-33  
fcFxpPortC3InFrames 5-32  
fcFxpPortC3InOctets 5-33  
fcFxpPortC3OutFrames 5-32  
fcFxpPortC3OutOctets 5-33  
fcFxpPortCapBbCreditMax 5-35  
fcFxpPortCapBbCreditMin 5-35  
fcFxpPortCapClass2SeqDeliv 5-37  
fcFxpPortCapClass3SeqDeliv 5-37  
fcFxpPortCapCos 5-36  
fcFxpPortCapFcphVersionHigh 5-34  
fcFxpPortCapFcphVersionLow 5-34  
fcFxpPortCapHoldTimeMax 5-38

fcFxpPortCapHoldTimeMin 5-38  
fcFxpPortCapIntermix 5-36  
fcFxpPortCapRxDataFieldSizeMax 5-35  
fcFxpPortCapRxDataFieldSizeMin 5-36  
fcFxpPortCapStackedConnMode 5-37  
fcFxpPortCapTable 5-34  
fcFxpPortClass2SeqDeliv 5-10  
fcFxpPortClass2SeqDelivAgreed 5-19  
fcFxpPortClass3SeqDeliv 5-10  
fcFxpPortClass3SeqDelivAgreed 5-19  
fcFxpPortConnectedNxPort 5-20  
fcFxpPortCosSuppAgreed 5-18  
fcFxpPortCosSupported 5-9  
fcFxpPortDelimiterErrors 5-23  
fcFxpPortEdtov 5-9  
fcFxpPortFcphVersionAgreed 5-17  
fcFxpPortFcphVersionHigh 5-7  
fcFxpPortFcphVersionLow 5-7  
fcFxpPortHoldTime 5-11  
fcFxpPortID 5-11  
fcFxpPortIntermixSuppAgreed 5-18  
fcFxpPortIntermixSupported 5-9  
fcFxpPortInvalidCrcs 5-23  
fcFxpPortInvalidTxWords 5-22  
fcFxpPortLinkFailures 5-21  
fcFxpPortLinkReseatIns 5-24  
fcFxpPortLinkResetOuts 5-24  
fcFxpPortName 5-6  
fcFxpPortNxPortBbCredit 5-17  
fcFxpPortNxPortName 5-20  
fcFxpPortNxPortRxDataFieldSize 5-17  
fcFxpPortOlsIns 5-24  
fcFxpPortOlsOuts 5-25  
fcFxpPortOperMode 5-12  
fcFxpPortPhysAdminStatus 5-14  
fcFxpPortPhysEntry 5-14  
fcFxpPortPhysLastChange 5-16  
fcFxpPortPhysOperStatus 5-15  
fcFxpPortPhysRttov 5-16  
fcFxpPortPhysTable 5-14  
fcFxpPortPrimSeqProtoErrors 5-22  
fcFxpPortRatov 5-8  
fcFxpPortRxBufSize 5-8  
fcFxpPortSigLosses 5-22  
fcFxpPortStackedConnMode 5-10  
fcFxpPortStackedConnModeAgreed 5-19  
fcFxpPortSyncLosses 5-21  
Fx Port Fabric Login Table 5-16  
FxPort Configuration Table 5-6  
FxPort Physical Level Table 5-13

## G

Groups in MIB-II 3-1

## I

ICMP Group 3-27  
 icmpInAddrMaskReps 3-31  
 icmpInAddrMasks 3-30  
 icmpInDestUnreachs 3-28  
 icmpInEchoReps 3-29  
 icmpInEchos 3-29  
 icmpInErrors 3-28  
 icmpInMsgs 3-27  
 icmpInParmProbs 3-28  
 icmpInRedirects 3-29  
 icmpInSrcQuenchs 3-29  
 icmpInTimeExcds 3-28  
 icmpInTimestampReps 3-30  
 icmpInTimestamps 3-30  
 icmpOutAddrMaskReps 3-34  
 icmpOutAddrMasks 3-34  
 icmpOutDestUnreachs 3-31  
 icmpOutEchoReps 3-33  
 icmpOutEchos 3-33  
 icmpOutErrors 3-31  
 icmpOutMsgs 3-31  
 icmpOutParmProbs 3-32  
 icmpOutRedirects 3-33  
 icmpOutSrcQuenchs 3-32  
 icmpOutTimeExcds 3-32  
 icmpOutTimestampReps 3-34  
 icmpOutTimestamps 3-33  
 ifAdminStatus 3-7  
 ifDescr 3-5  
 ifIndex 3-5  
 ifInDiscards 3-9  
 ifInErrors 3-9  
 ifInNUcastPkts 3-8  
 ifInOctets 3-8  
 ifInUcastPkts 3-8  
 ifInUnknownProtos 3-9  
 ifLastChange 3-7  
 ifMtu 3-6  
 ifNumber 3-4  
 ifOperStatus 3-7  
 ifOutDiscards 3-10  
 ifOutErrors 3-11  
 ifOutNUcastPkts 3-10  
 ifOutOctets 3-10  
 ifOutQLen 3-11  
 ifOutUcastPkts 3-10  
 ifPhysAddress 3-7  
 ifSpecific 3-11  
 ifSpeed 3-6  
 ifType 3-6  
 Info 1-3, 4-76  
 Interfaces Group 3-4

Interfaces Table 3-5  
 IP Address Table 3-19  
 IP Address Translation Table 3-25  
 IP Group 3-13  
 IP Routing Table 3-21  
 ipAddrTable 3-19  
 ipAdEntAddr 3-19  
 ipAdEntBcastAddr 3-20  
 ipAdEntIfIndex 3-20  
 ipAdEntNetMask 3-20  
 ipAdEntReasmMaxSize 3-21  
 ipDefaultTTL 3-14  
 ipForwarding 3-13  
 ipForwDatagrams 3-15  
 ipFragCreates 3-19  
 ipFragFails 3-19  
 ipFragOKs 3-18  
 ipInAddrErrors 3-14  
 ipInDelivers 3-16  
 ipInDiscards 3-16  
 ipInHdrErrors 3-14  
 ipInReceives 3-14  
 ipInUnknownProtos 3-15  
 ipNetToMediaIfIndex 3-25  
 ipNetToMediaNetAddress 3-26  
 ipNetToMediaPhysAddress 3-26  
 ipNetToMediaType 3-26  
 ipOutDiscards 3-17  
 ipOutNoRoutes 3-17  
 ipOutRequests 3-16  
 ipReasmFails 3-18  
 ipReasmOKs 3-18  
 ipReasmReqds 3-18  
 ipReasmTimeout 3-17  
 ipRouteAge 3-24  
 ipRouteDest 3-21  
 ipRouteIfIndex 3-21  
 ipRouteInfo 3-25  
 ipRouteMask 3-24  
 ipRouteMetric1 3-22  
 ipRouteMetric2 3-22  
 ipRouteMetric3 3-22  
 ipRouteMetric4 3-23  
 ipRouteMetric5 3-25  
 ipRouteNextHop 3-23  
 ipRouteProto 3-24  
 ipRouteTable 3-21  
 ipRouteType 3-23  
 ipRoutingDiscards 3-27

## L

Link Table 4-44

## M

Management information bases 3-1  
Manager 1-1  
Mark 4-76  
MIB Definitions 5-1  
MIB-II 3-1  
Module Table 5-3

## N

Notify 1-3, 4-76

## P

Port Statistics Table 4-50  
Port Table 4-26

## R

Revision Table 4-18  
revisionNumber 4-2

## S

SANbox Manager 2-3  
Sensor Table 4-20  
Simple Name Server Table 4-69  
Simple Network Management Protocol 1-1  
SNMP 1-1  
SNMP Group 3-49  
snmpEnableAuthenTraps 3-57  
snmplnASNParseErrs 3-51  
snmplnBadCommunityNames 3-50  
snmplnBadCommunityUses 3-50  
snmplnBadValues 3-52  
snmplnBadVersions 3-50  
snmplnGenErrs 3-52  
snmplnGetNexts 3-53  
snmplnGetRequests 3-53  
snmplnGetResponses 3-54  
snmplnNoSuchNames 3-51  
snmplnPkts 3-49  
snmplnReadOnlys 3-52  
snmplnSetRequests 3-54  
snmplnTooBig 3-51  
snmplnTotalReqVars 3-53  
snmplnTotalSetVars 3-53  
snmplnTraps 3-54  
snmpOutBadValues 3-55  
snmpOutGenErrs 3-56  
snmpOutGetNexts 3-56  
snmpOutGetRequests 3-56  
snmpOutGetResponses 3-57  
snmpOutNoSuchNames 3-55  
snmpOutPkts 3-50  
snmpOutSetRequests 3-56

snmpOutTooBig 3-55  
snmpOutTraps 3-57  
SNMPv1 1-1  
SNMPv2c 1-1  
SNMPv3 1-1  
Status Group 5-11  
statusChangeTime 4-4  
sysContact 3-3  
sysDescr 3-2  
sysLocation 3-4  
sysName 3-3  
sysObjectID 3-2  
sysServices 3-4  
System Group 3-1  
systemURL 4-3  
sysUpTime 3-2

## T

TCP Connection Table 3-38  
TCP Group 3-35  
tcpActiveOpens 3-36  
tcpAttemptFails 3-37  
tcpConnLocalAddress 3-39  
tcpConnLocalPort 3-39  
tcpConnRemAddress 3-40  
tcpConnRemPort 3-40  
tcpConnState 3-39  
tcpCurrEstab 3-37  
tcpEstabResets 3-37  
tcpInErrs 3-40  
tcpInSegs 3-38  
tcpMaxConn 3-36  
tcpOutRsts 3-41  
tcpOutSegs 3-38  
tcpPassiveOpens 3-36  
tcpRetransSegs 3-38  
tcpRtoAlgorithm 3-35  
tcpRtoMax 3-35  
tcpRtoMin 3-35  
Telnet 2-2  
Transmission Group 3-49  
trap severity levels 1-3, 4-76  
Trap Table 4-76  
trapRegFilter 4-78  
trapRegIpAddress 4-77  
trapRegPort 4-77  
trapRegRowState 4-78

## U

UDP Group 3-41  
UDP Listener Table 3-42  
udpInDatagrams 3-41  
udpInErrors 3-41

---

udpLocalAddress 3-42  
udpLocalPort 3-42  
udpNoPorts 3-41  
udpOutDatagrams 3-42  
Unknown 1-3, 4-76  
uNumber 4-3

## **W**

Warning 1-3, 4-76