CHAPTER 3

Using the MIB

This chapter describes the organization and contents of the StrataView Plus Proxy Agent Management Information Base (MIB). The Proxy Agent MIB is structured as two separate MIBs:

- The Network MIB contains the contents of the events log in the database and network configuration information. This MIB is supported by the RtmProxy. A complete listing of the Network MIB is provided in Appendix A.
- The Service MIB, a consolidated MIB for port and connection services, contains the frame relay and ASI (ATM Service Interface) connection information for both local and remote end-points and frame relay port information. The Connection Service MIB is supported by the ConnProxy, and the Port Service MIB is supported by the PortProxy. A complete listing of the Service MIB is provided in Appendix B.

This chapter also provides illustrations of how a connection is created with SNMP SET Requests and how an SNMP Manager can assemble MIB objects to construct and display the network topology. To fully appreciate the information in this chapter, you must have some knowledge of the SNMP MIB conventions and standards.

MIB Organization

The StrataView Plus Proxy Agent Network and Service MIBs are organized sets of objects, each of which contains a piece of information regarding the IPX/BPX/AXIS network and services.

The SNMP community string is treated differently on these MIBs. It is used for authentication in the Service MIB, and it is used as part of the instance in the Network MIB. The community string for identifying a node is the Domain.Node (e.g., Network1.node3) in the Node Table in the Network MIB.

Each object is assigned a unique identifier within the MIB. Objects are accessed by the SNMP Manager by issuing GET and GETNEXT commands which specify the object's unique identifier. The Proxy Agent obtains the value of the specified object and transmits it to the SNMP manager.

In the Network MIB, the SNMP Manager has read-only access to the objects with the exception of two tables, namely, the EventFilter Table and the trapsConfig Table. In the Service MIB, the following tables are read-write:

- frFndPt Table
- atmEndPt Table
- connection Table
- frPortsCfg Table.

If a single piece of information contained in a single object is required, a simple GET of that object by the SNMP Manager is all that is necessary to retrieve the information. To obtain more complex information about the network requires the retrieval of several objects and the interpretation of their values. This is the situation when the SNMP Manager needs to construct and display the network topology.

Network MIB Tables and Groups

The Network MIB, which is supported by the sub-agent RtmProxy, is organized into the following major tables and groups:

Events Control Group	The Events Control Group is the highest level entity. It has attributes which control event processing such as the database polling rate.
A Maintenance Log Table	This table contains a circular queue of the log record objects currently in the database. The circular queue can hold up to 8192 log records.
Maintenance Log Filter	This group contains a set of attributes to permit filtering the output of GetNext operations on the Maintenance Log Table. The attributes enable filtering on node name, network name, log severity, event time, and a user specified maximum number of log records.
An Event Filter MIB Table	This table contains a boolean expression which is used to match new log records in the database. When a new log record occurs that matches the boolean expression, an SNMP TRAP is generated. The boolean expression is the mechanism for filtering which log records generate traps in addition to being logged in the Maintenance Log Table. Use a community string of “public” to access the events log MIB.
A Network Table	This table contains entries for each IPX/BPX/AXIS network being managed by the StrataView system.
A Node Table	This table contains entries for each node being managed by the StrataView system. Each entry contains ID, configuration and status information about the node.
A Node Group	The Node group contains attributes from the Informix data base which apply to an individual node (for example, status).
A Trunk MIB Table	This table contains configuration information for each trunk in the node.
Trunk RTC Table	This table contains entries for realtime counter maintained for each trunk in the node.
A Circuit Line MIB Table	This table contains configuration and summary statistical information for each circuit line in the node.
A Circuit Line RTC Table	This table contains entries for realtime counter maintained for each circuit line in the node.

A Frame Relay Port Table	This table contains configuration and summary statistical information for each frame relay port in the node.
A Frame Relay RTC Table	This table contains entries for realtime counter maintained for each frame relay port in the node.
A Connection Table	This table contains configuration and summary statistical information for each frame relay end point in the node.
A Connection RTC Table	This table contains entries for realtime counter maintained for each connection in the node.
trapConfig Table	The SNMP manger must add an entry in this table to get any traps from the Proxy.
trapUpLoad Table	The SNMP Manger can do a GET Request on this table to recover a lost trap from the Proxy.
Trap Definitions	Defines the four status alarm traps that are generated by the Proxy Agent. There is a trap for trunk alarms circuit line alarms, frame relay port alarms, and connection alarms.

Service MIB Tables

The Service MIB is a consolidated MIB for connections and port services.

The Connection MIB, which is supported by the sub-agent ConnProxy, is organized within the following major tables:

Frame Relay End-Point Table	This table contains the Frame Relay end points of the connections in AXIS and IPX nodes. Frame Relay end points are either local or remote end points of a connection.
ATM End-Point Table	This table contains the ATM end points, which are the BPX routing segment end-points of a Frame Relay connection. It also contains ASI End Points of the SIW (Service Internetworking) connections.
Connection Table	This table contains end-to-end connections, and parameters like LocalEndPt, RemoteEndPt, ClassOfService, Preferred Route, etc.
Segment Table	This table contains information about all the segments of a connection. Given an end-point in the connection, you can find the other end point of the segment and also the local end-point of the next segment in the connection.
Error Table	This table contains all the errors encountered in SET requests, indexed by the Request ID. This table is also indexed on the SNMP manager’s address, but is transparent to the user.
A Ports Info Table	This table contains configuration information about AXIS and IPX frame relay ports.

The Port MIB, which is supported by the sub-agent PortProxy, is organized within the following major tables:

Ports Info Table	This read-only table contains the main attributes of a Frame Relay or ATM port. (Currently only Frame Relay ports are supported.)
Ports Config Table	This table is used for configuring IPX or AXIS Frame Relay Ports. Ports can be added, modified, or deleted. There are some restrictions on the configuration depending on whether the port is from an IPX or AXIS shelf. (These restrictions are described in the MIB.)
Error Table	This table contains all the errors encountered in SET requests, indexed by the Request ID. This table is also indexed on the SNMP manager's address, but is transparent to the user

Network Topology

In order to construct and display the network topology, the SNMP Manager needs to assemble a number of objects from various parts of the Network and Service MIB files. The process starts at the IPX/BPX/AXIS network level and follows with determining the active nodes within the network. For each node in the network, the active trunks are determined together with their end node IDs and IPX/BPX/AXIS line numbers. The alarm status of the nodes and the trunks is also determined.

With this information, a topological display of the network can be constructed and displayed by the SNMP Manager. With knowledge of the alarm status, the ability to color code the nodes and the trunks in order to indicate their alarm conditions is provided.

Network ID

The network ID is accessed through the Network MIB Network Table which contains an entry for each network in the system. Each entry in this table contains the following objects for each IPX network being managed by StrataView Plus:

- Network ID (unique network ID assigned by StrataView Plus)
- Network IPX ID (the IPX network ID)
- Network name (user defined network name)

Issuing GET commands for the Network Table entries obtains the Network ID for the network under consideration.

Nodes

The active nodes in the selected network can be determined by examining the entries in the Network MIB Node Group. This Node Group contains an entry for each node in the network and each entry contains the following objects for the node:

- Node network name (the name of the network to which the node is attached)
- Node name (IPX Node name)

The nodes in the network under consideration can be determined by examining each node table entry and determining which nodes are contained in the network.

By this process, the number and identification of the nodes in the network is established. The interconnecting trunks still need to be determined to construct the topology trunks.

The trunks connecting the nodes in the selected network can be determined by examining the entries in the Trunk Table. The Trunk Table contains an entry for each trunk in the system and each entry contains the following objects for the trunk:

- Trunk local slot
- Trunk local port
- Trunk local line
- Trunk card type
- Trunk interface
- Trunk line load
- Trunk remote node ID
- Trunk remote line number
- Trunk remote slot
- Trunk remote port
- Trunk alarm state
- Trunk comment
- Trunk active state
- Trunk status
- Trunk Statistical Reserve

The trunks in the network can be determined by examining each trunk entry and determining which trunks are connected to which nodes in the network (from the local and remote line number, slot and port fields and the remote node ID). The table entry also shows which trunks are active (from trunkActive state field) and the status of trunk alarms.

Note that each Trunk entry contains the Node ID of the remote node at the other end of the trunks. Using this information the topology of the network can be constructed Figure 3-1shows an example network consisting of three nodes named alpha (nodeID=0), beta (nodeID=1) and gamma (nodeID=2). Each node is connected to the other two by packets lines to form a triangular network

Figure 3-1 Example IPX network

Assuming the networkIpxId is 01, the entries in the node group can be read to see which nodes have the **nodeIpxNetId** value of 01. This will yield three nodes as follows:

nodeID=0	nodeName=alpha	alarmstate=1(clear)	nodeActive=2(active)
nodeID=1	nodeName=beta	alarmstate=1(clear)	nodeActive=2(active)
nodeID=2	nodeName=gamma	alarmstate=1(clear)	nodeActive=2(active)

The entries in the Trunk table can be read to see which trunks have the trunkRemoteNodeId with a value of 0, 1 or 2. This will yield three trunks as follows (as an example called 196613, 196615, and 196617)

:

trunkId=196613

trunkLocalNodeId=0	trunkNumber=14
trunkRemNodeId=2	trunkRemNumber=13
trunkAlarmState=clear	trunkActive=2 (active)

trunkId=196615

trunkLocalNodeId=0	trunkNumber=13
trunkRemNodeId=2	trunkRemNumber=12
trunkAlarmState=clear	trunkActive=2 (active)

trunkId=196615

trunkLocalNodeId=1 trunkNumber=12

trunkRemNodeId=0 trunkRemNumber=13

trunkAlarmState=clear trunkActive=2 (active)

trunkId=196617

trunkLocalNodeId=1 trunkNumber=13

trunkRemNodeId=2 trunkRemNumber=12

trunkAlarmState=clear trunkActive=2 (active)

trunkId=196613

trunkLocalNodeId=2 trunkNumber=13

trunkRemNodeId=0 trunkRemNumber=14

trunkAlarmState=clear trunkActive=2 (active)

trunkId=196617

trunkLocalNodeId=2 trunkNumber=12

trunkRemNodeId=1 trunkRemNumber=13

trunkAlarmState=clear trunkActive=2 (active)

With the above information, the network diagram shown in Figure 3-1 can be constructed and the node and trunk elements can be displayed in color to indicate alarm status.

The details of the network can be increased in similar fashion by adding Circuit Line information from the Circuit Line Table, Frame Relay information from the Frame Relay Port Table and Connection information from the Connection Table.

Event Log

StrataView Plus maintains a network event log which is accessible to the SNMP Manager through the Maint Log Table in the MIB. Events are generated by the IPX/BPX/AXISIPX/BPX/AXIS network and transmitted to StrataView Plus as they occur.

Each event is a separate record in a circular log which can store up to 8192 events. As new events are added to the log an event moves around the circular file and is eventually overwritten. It is the responsibility of the SNMP Manager to read the events before they are overwritten.

In the MIB, each event is an entry in the Maint Log Table which contains the following objects:

logIndex	A unique identifier of the log record used to reference a log entry
logNetwork	The name of the network generating the log entry
logNodeName	The name of the node generating the log entry
logGmtDate	The time and date of the event reported by the IPX
logSeverity	The severity of this log record
logMsg	An ASCII message associated with this log record

Because the number of records in the Maint Log table is large and access may be relatively slow, the ability to filter based on node name, network name, time and severity is provided for GetNext operations. In addition, the results of the filter may be limited to a user settable window size which further limits the SNMP view of the Maint Log table.

The Main Log log filter mechanisms are controlled by attributes in the Main Log Filter table. This table contains a set of filtering attributes which are applied to qualify the event in the following order.

maintLogFilterTimeMin	If other than zero, a minimum value for time to qualify the log entry.
maintLogFilterTimeMax	If other than zero, a maximum value for time to qualify the log entry.
maintLogFilterNetworkName	If other than zero, this network name is used to qualify log entries.
maintLogFilterNodeName	If other than zero, this node name is used to qualify log entries.
maintLogFilterSeverity	A character string from 0 to 5 characters. If other than zero, the string value is used to qualify log entries.
maintLogFilterWindow	If other than zero, the window value is used to limit the view of records in the Main Log table to the number specified for this attribute. If, after applying all the other filter attributes, the remaining set of Main Log entries exceeds the value of the maintLogFilterWindow then, the set of entries that can be viewed is limited to the top N based on the logIndex value (where N is the window value).

Events added to the log may generate SNMP traps provided they satisfy trap filter conditions. Events that do not generate traps are merely recorded as an entry in the Maintenance Log table and are accessed by the SNMP Manager through normal GET and GET NEXT commands.

Traps

The Event Trap mechanism is provided to alert the SNMP Manager of important events without the SNMP Manager having to read the whole event log. When an event is added to the log, its logSeverity and logMsg fields are examined to see if they satisfy a filter condition.

The trap filter mechanism uses Boolean expressions that are applied to the logSeverity and logMsg fields of messages as they are added to the Event Log. If these fields match one or more of the Boolean expressions, an SNMP TRAP is generated.

Periodically, the Proxy Agent polls the Event Log MIB and filter for traps and all events flagged as traps are sent to the SNMP Manager.

The Boolean expressions are stored in the Event Filtering Table as a local configuration file. The polling rate at which the SNMP Manager polls for traps is also stored as a local configuration file. These two files are created by the SNMP Manager which has Read/Write access. This is different from all the other MIB tables and files for which the SNMP Manager has Read Only access.

Alarms generated by the Asynchronous Alarm Generator are also forwarded as traps. The Proxy also forwards the traps generated by an AXIS.

Each entry in the Event Filtering Table contain the following objects:

eventFilterIndex	A unique identifier for the entry in the table. This identifier is provided by the SNMP manager with a SET command containing a new value for eventFilterIndex. This command establishes a new row in the table. Thus multiple rows (multiple sets of filter parameters) may be maintained in the table with each row referenced by its index number. With multiple row tables, the row that is used for filtering is the row with the eventFilterStatus set to active.
eventFilterStatus	Used to indicate whether the entry is invalid or active.
eventFilterSeverity	Specifies the severity of log records in order to generate a trap
eventFilterSubstring	Specifies an ASCII substring that must appear in the log record's msg field in order to generate a trap

Statistics

A wide variety of network statistics are maintained as realtime counters in the MIBs. Each statistic counter is an unsigned 32-bit number which indicates the number of instances the statistical event has occurred since the counter was last reset to zero. Counters are reset to zero after an NPC rebuild, an NPC switchover or when the 32-bit value exceeds the maximum value for the field and flips over to zero.

MIB objects representing the statistics are organized by type and maintained in the appropriate table. Trunk counters are maintained in the Trunk RTC Table, Frame relay counters are maintained in the Frame Relay RTC Stat Table and so on. The retrieval of realtime counters can be time consuming (as much as 10 seconds per counter), Get-Next walks through the realtime counter tables should be avoided.

The following realtime counters are available from the MIBs. A short description of each statistic type is provided in the formal MIB descriptions in Appendices A and B.

Trunk Counters (Included in the Trunk RTC Table Entries)

trunkRTCLocalSlot	INTEGER,
trunkRTCLocalPort	INTEGER,
trunkRTCBipolarViolations	Counter,
trunkRTCFrameSlips	Counter,
trunkRTCOutOfFrames	Counter,
trunkRTCLossOfSignal	Counter,
trunkRTCFrameBitErrors	Counter,
trunkRTCCrcErrors	Counter,
trunkRTCPktOutOfFrames	Counter,
trunkRTCPktCrcErrors	Counter,
trunkRTCBadClockErrors	Counter,
trunkRTCVoicePktsDropped	Counter,
trunkRTCTimeStampedPktsDropped	Counter,
trunkRTCNonTimeStampedPktsDropped	Counter,
trunkRTCHighPriorityPktsDropped	Counter,
trunkRTCBurstyDataPktsDropped	Counter,
trunkRTCMulticastPktsDropped	Counter,
trunkRTCVoicePktsXmitted	Counter,
trunkRTCTimeStampedPktsXmitted	Counter,
trunkRTCNonTimeStampedPktsXmitted	Counter,
trunkRTCHighPriorityPktsXmitted	Counter,
trunkRTCBurstyDataPktsXmitted	Counter,
trunkRTCMulticastPktsXmitted	Counter,
trunkRTCPktsXmitted	Counter,
trunkRTCTxBurstyDataAClpPktsDropped	Counter,
trunkRTCTxBurstyDataBClpPktsDropped	Counter,
trunkRTCBurstyDataAEfcnPktsTx2Line	Counter,
trunkRTCBurstyDataBEfcnPktsTx2Line	Counter,

trunkRTCBurstyDataAClpPktsTx2Line	Counter,
trunkRTCBurstyDataBClpPktsTx2Line	Counter,
trunkRTCAtmCellHeaderHecErrors	Counter, *
trunkRTCTxVoiceCellsDropped	Counter, *
trunkRTCTxTimeStampCellsDropped	Counter, *
trunkRTCTxNonTStampCellsDropped	Counter, *
trunkRTCTxHighPriorityCellsDropped	Counter, *
trunkRTCTxBurstyDataACellsDropped	Counter, *
trunkRTCTxBurstyDataBCellsDropped	Counter, *
trunkRTCVoiceCellsTx2Line	Counter, *
trunkRTCTimeStampCellsTx2Line	Counter, *
trunkRTCNonTimeStampCellsTx2Line	Counter, *
trunkRTCHighPriorityCellsTx2Line	Counter, *
trunkRTCBurstyDataACellsTx2Line	Counter, *
trunkRTCBurstyDataBCellsTx2Line	Counter, *
trunkRTCTotalCellsTx2Line	Counter, *
trunkRTCTxBurstyDataAClpCellsDropped	Counter, *
trunkRTCTxBurstyDataBClpCellsDropped	Counter, *
trunkRTCBurstyDataAEfcnCellsTx2Line	Counter, *
trunkRTCBurstyDataBEfcnCellsTx2Line	Counter, *
trunkRTCPlcpOutOfFrames	Counter, *

* Realtime counter applies to ATM trunks only.

Circuit Line Counters (Included in the Circuit Line RTC Table Entries)

cirLineRTCLineNumber	INTEGER,
cirLineRTCBipolarViolations	Counter,
cirLineRTCFrameSlip	Counter,
cirLineRTCOutOfFrames	Counter,
cirLineRTCLossesOfSignal	Counter,
cirLineRTCFrameBitErrors	Counter,
cirLineRTCCrcErrors	Counter,
cirLineRTCOutOfMultiFrames	Counter,
cirLineRTCAllOnesInTimeslot16	Counter

Frame Relay Counters (Included in the Frame Relay Port RTC Table Entries)

frpRTCSlot	INTEGER,
frpRTCPort	INTEGER,
frpRTCFramesRcvd	Counter,
frpRTCFramesXmitted	Counter,
frpRTCBytesRcvd	Counter,
frpRTCBytesXmitted	Counter,
frpRTCFramesXmittedWithFECN	Counter,
frpRTCFramesXmittedWithBECN	Counter,
frpRTCFramesRcvdCrcErrors	Counter,
frpRTCFramesRcvdInvalidFormat	Counter,
frpRTCFramesRcvdAlignmentErrors	Counter,
frpRTCFramesRcvdIllegalLen	Counter,
frpRTCDmaOverruns	Counter,
frpRTCLmiStatusEnquires	Counter,
frpRTCLmiStatusXmitRate	Counter,
frpRTCLmiStatusUpdateRate	Counter,
frpRTCLmiInvalidStatusEnquires	Counter,

frpRTCLmiLinkTimeoutErrors	Counter,
frpRTCLmiKeepaliveSequenceErrors	Counter,
frpRTCFramesRcvdUndefDlciErrors	Counter,
frpRTCXmitStatusEnquiry	Counter,
frpRTCRxStatusCounter	Counter,
frpRTCAsyncStatusCounter	Counter,
frpRTCBadSequenceNumberCount	Counter,
frpRTCTxProtocolTimeOutCount	Counter,
frpRTCCLLMFramesTx	Counter,
frpRTCCLLMBytesTx	Counter,
frpRTCCLLMFramesRx	Counter,
frpRTCCLLMBytesRx	Counter,
frpRTCCLLMFailures	Counter,
frpRTCRxDEFramesDiscarded	Counter

Connection Counters (Included in the Connection RTC Table Entries)

connRTCSlot	INTEGER,
connRTCChannel	INTEGER,
connRTCDLCI	INTEGER,
connRTCRcvdFrames	Counter,
connRTCRcvdFramesDiscarded	Counter,
connRTCXmitFrames	Counter,
connRTCXmitFramesDiscarded	Counter,
connRTCRcvdPkts	Counter,
connRTCRcvdPktsDiscarded	Counter,
connRTCXmitPkts	Counter,
connRTCXmitPktsProjected	Counter,
connRTCXmitPktsSupervisory	Counter,
connRTCRcvdBytes	Counter,

connRTCSlot	INTEGER,
connRTCRcvdBytesDiscarded	Counter,
connRTCXmitBytes	Counter,
connRTCXmitBytesDiscarded	Counter,
connRTCSecondsV25ModemOn	Counter,
connRTCSecondsDsiEnabled	Counter,
connRTCSecondsOffHook	Counter,
connRTCSecondsInService	Counter,
connRTCXmitFramesWithFECN	Counter,
connRTCXmitFramesWithBECN	Counter,
connRTCRxSupervisoryPkts	Counter,
connRTCCongestedMinutes	Counter,
connRTCFramesRxWithDE	Counter,
connRTCFramesTxWithDE	Counter,
connRTCFramesDiscardedWithDE	Counter,
connRTCBytesRxWithDE	Counter,
connRTCFramesRxExcessCir	Counter,
connRTCBytesRxExcessCir	Counter

Creating Connections and Ports

This section provides some examples for:

- Creating a Connection
- Creating a Port.

Creating a Connection

Creating a connection involves one SET Request, which creates two end points and one connection entry.

To create end points and a connection entry, you SET a minimum of five attributes in the Frame Relay End Point table (frEndPtTable) and Connection table (connectionTable).

The only mandatory attribute that must be set in the FrameRelay End Point Table is frEndPtRowStatus, and the value must be createAndGo. Since frEndPtNodeName, frEndPtIfShelf, frEndPtSlot, frEndPtLine, frEndPtPort, and frEndPtDlci are the indices. They need to be supplied as part of the frEndPtRowStatus OID.

The SET Request must set attributes in connectionTable, connectionLocalEndPt, connectionRemoteEndPt, and connectionRowStatus (createAndGo). (You must use the special connection index value *0*.)

If required, other read-write attributes from the frEndPtTable and connectionTable can also be included as part of the same SET Request. If other attributes are not supplied, default values will be considered.

Note For creating ASI end points, the atmEndPtTable is used.

Connection Example

The following example assumes a 2 segment IPX (router) to AXIS (feeder) connection, as shown in Figure 3-2.

Figure 3-2 Sample Connection

Setup

The setup of the two nodes is as follows:

Node1=8.110.109.115.105.112.120.49.48
(Encoded form of “nmsipx10”)

Shelf1=0

Slot1=6
Line1=0
Port1=1
Dlci1=200

Node2=8.110.109.115.98.112.120.48.51
(Encoded form of “nmsbxp03”)

Shelf2=7.65.88.73.83.50.52.53
(Encoded form of “AXIS245”)

Slot2=6
Line2=1
Port2=1
Dlci2=200

SET Request

The SET Request must contain the following five OIDs:

Oid : 1.3.6.1.4.1.351.3.4.1.10.$Node1.$Shelf1.$Slot1.$Line1.$Port1.$Dlci1
name : frEndPtRowStatus
Type : Integer
Value : createAndGo (4)

Oid : 1.3.6.1.4.1.351.3.4.1.10.$Node2.$Shelf2.$Slot2.$Line2.$Port2.Dlci2
name : frEndPtRowStatus
Type : Integer
Value : CreateAndGo(4)

Oid : 1.3.6.1.4.1.351.3.3.1.2.0
name : connectionLocalEndPt
Type : Oid
Value : 1.3.6.1.4.1.351.3.4.1.1.$Node1.$Shelf1.$Slot1.$Line1.$Port1.$Dlci1

Oid : 1.3.6.1.4.1.351.3.3.1.3.0
name : connectionRemoteEndP
Type : Oid
Value : 1.3.6.1.4.1.351.3.4.1.1.$Node2.$Shelf2.$Slot2.$Line2.$Port2.Dlci2

Oid : 1.3.6.1.4.1.351.3.3.1.6.0
name : connectionRowStatus
Type : Integer
Value : CreateAndGo(4)

Other read-write attributes can also be included in the same SET request, such as:

```
Oid     : 1.3.6.1.4.1.351.3.4.1.11.$Node1.$Shelf1.$Slot1.$Line!.$Port1.$Dlci1
name    : frEndPtMIR
Type    : Integer
Value   : 2400

Oid     : 1.3.6.1.4.1.351.3.4.1.15.$Node2.$Shelf2.$Slot2.$Line2.$Port2.$Dlci2
name    : frEndPtVcQSize
Type    : Integer
Value   : 5000

Oid     : 1.3.6.1.4.1.351.3.3.1.10.0
name    : connectionTrkAvoidZCS
Type    : Integer
Value   : true (2)

Oid     : 1.3.6.1.4.1.351.3.3.1.10.0
name    : connectionClassOfService
Type    : Integer
Value   : 10
```

Note Default values will be applied to the read-write attributes that are not supplied with this SET Request.

Creating a Port

You need only a single SET Request with one mandatory object, frPortsCfgRowStatus, to create a port.

The following example assumes the following:

Node=8.110.109.115.98.112.120.48.51
(Encoded form of “nmsbpx03”)

Shelf=7.65.88.73.83.50.52.53
(Encoded form of “AXIS245”)

Slot=6
Line=2
Port=1

To create this port, your SET Request must have the following object:

```
Oid     : 1.3.6.1.4.1.351.4.2.1.6.$Node.$Shelf.$Slot.$Line.$Port
name    : frPortsCfgRowStatus
Type    : Integer
Value   : add(1)
```

Other read-write objects could also be included in the same SET Request, such as:

```
Oid     : 1.3.6.1.4.1.351.4.2.1.10.$Node.$Shelf.$Slot.$Line.$Port
name    : frPortsCfgPortSpeed
Type    : Integer
Value   : 224

Oid     : 1.3.6.1.4.1.351.4.2.1.11.$Node.$Shelf.$Slot.$Line.$Port
name    : frPortsCfgDs0ChSpeed
Type    : Integer
Value   : s56k(1)
```