Technical Document

Niagara$^{AX-3.5}$ Snmp Driver Guide

March 23, 2011

Niagara$^{AX-3.5}$ Snmp Driver Guide

Contents

PREFACE

Preface

Document Change Log

Updates (changes/additions) to this Niagara$^{AX-3.5}$ *Snmp Driver Guide* document are listed below.

- Initial Publication of the Niagara$^{AX-3.5}$ version, March 23, 2011
  New document that reflects updates to the snmp module.

What's new in the Niagara$^{AX-3.5}$ Snmp Driver

The following list describes the major changes that have been made to the NiagaraAX Snmp Driver since the last version of this document was released.

- **MIB compiler improvements**
  The standard MIBs are now built into the snmp driver so you do not need to manually access them or point to them if you require them as dependent MIBs. See "About locally generated MIBs" on page 3-17.
- **Create custom MIBs**
  You can now create a custom MIB on the Local Device. The custom MIB will include entries for the current MIB objects as well as Export Tables. See "About locally generated MIBs" on page 3-17.
- **Snmp Servlet**
  You can now create a custom MIB using a station snmp servlet. See "About the SNMP Servlet" on page 3-21.
- **Table Object and Table Manager view**
  There is now a Table Object and a Table Manager view with a **`WalkMib`** button. See "About the Snmp Table Manager view" on page 3-24.
- **Export Table and Export Table Manager view**
  This new feature allows you to discover existing control points on your Local Device and add them to an export table. Points in the export table are then available for discovery by an SNMP Manager (client) device. See "About the Snmp Export Manager (SnmpExportTable)" on page 3-26.
- **Export point folders**
  Nested export folders are available using a button on the Snmp Export Manager view. These folder components can be nested and they adjust OIDs if points are moved between folders. Each folder is represented separately in the MIB as an input and output table, the same way that points in the Points folder are represented. See "About the Snmp Export Manager (SnmpExportTable)" on page 3-26.
- **Additional data types supported**
  Added support for enum and boolean MIB objects. See "snmp-SnmpBooleanExport" and "snmp-SnmpBooleanProxyExt", "snmp-SnmpBooleanExport", and "snmp-SnmpNetwork" on page 5-3.
- **Discovery interface updates**
  New and improved Discovery process that allows you to assign multiple dependent MIBs, if desired. See "About SNMP point discovery" on page 3-23.
- **Snmp Objects added with Object Manager view**
  New SnmpObjects are added to allow values to be exposed as any Snmp ASN data type instead of just the string type supported in the tables. See "About the Snmp Object Manager view" on page 3-27.

- **Alarm Table capacity option**
  An Alarm Table capacity property is available on the SnmpNetwork component. This allows you to specify a maximum number of alarms that are allowed to be stored in memory. See "Snmp Alarm Table Capacity" on page 3-5.
- **MIB path property**
  A MIB path property is now available directly on the NMP Device component so you can specify or modify the path to a custom MIB.

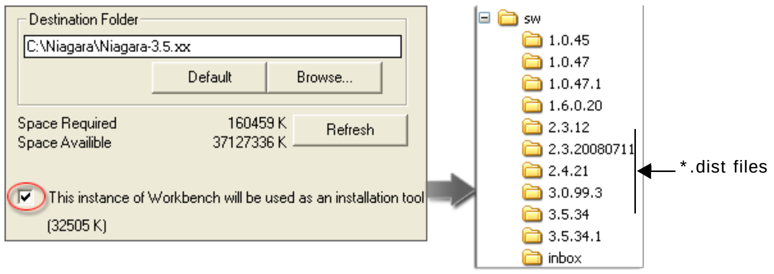
CHAPTER 1

SNMP Driver Installation

The following list describes the items that you need to consider for installation of the SNMP driver.

- **Licensing**
  To use the Niagara$^{AX-3.5}$ SNMP driver, you must have a target controller host that is licensed with the “SNMP” feature. In addition, other SNMP device limits or proxy point limits may exist in your license.
- **Workbench install tool option (“dist” files installed)**
  From your PC, use the Niagara Workbench 3.*5.nn* (or later) that was installed with the “installation tool” option selected, as shown in Figure 1-1. This option installs the needed distribution files (*.dist* files) for commissioning various models of remote controller platforms. If installed, the dist files are located under your Niagara installation directory under a “sw” subdirectory. For details, see “About your software database” in the *Platform Guide*.

Figure 1-1 *Installing Workbench installation tool option*


Destination Folder
C:\Niagara\Niagara-3.5.xx
Space Required 160459 K
Space Availible 37127336 K
[✓] This instance of Workbench will be used as an installation tool
(32505 K)

- **SNMP module**
  Apart from installing the 3.*5.nn* (or newer) version of the Niagara distribution in the controller, make sure to also install the *SNMP* module plus any specific (“private”) MIB files and required MIB dependency files.
  Note: *All “standard” MIB files are included as part of the SNMP module. The MIB files are used in discovering SNMP device data points.*
  Upgrade any modules shown as “out of date”. For instructions about updating your modules, see “Software Manager” in the *Platform Guide*.
  The remote controller is now ready for SNMP Network configuration in its running station, as described in “Configure the SnmpNetwork” on page 2-1.

Note: *Basic procedures for using the online features of the driver, including discovery of online SNMP devices and points, is discussed in “Create Snmp proxy points” on page 2-3.*

CHAPTER 2

SNMP Quick Start

This section provides procedures and descriptions that are commonly required and used with the NiagaraAX SNMP driver in typical online scenarios. Like other NiagaraAX drivers, you can do most configuration from special "manager" views and property sheets using Workbench. The procedures are grouped in the following sections:

- Configure the SnmpNetwork
- "Create a MIB" on page 2-2
- Create Snmp proxy points

Configure the SnmpNetwork

To configure the SnmpNetwork, perform the following main tasks:

- Adding an SnmpNetwork
- Designing an Snmp network application

Adding an SnmpNetwork

When you add an SnmpNetwork to a station, a "local device" ( ) component comes with the SnmpNetwork component. This local device (also referred to as an "Snmp Agent") allows you to expose data to *remote* Snmp sources (managers). Only one local device is allowed under the SnmpNetwork and is pre-configured as "enabled", by default.

To add an SnmpNetwork to the station

Use the following procedure to add an SnmpNetwork component under the station Drivers container.

To add an SnmpNetwork in the station:

Step 1 Double-click the station **`Drivers`** container. The About the Driver Manager view appears in the view pane.

Step 2 Click the **`New`** button to bring up the **`New DeviceNetwork`** dialog box. For more details, see Driver Manager New and Edit in the *NiagaraAX-3.x Drivers Guide*.

Step 3 Select "`SnmpNetwork`," number to add: `1` and click **`OK`**.

Note: *You can add only one SnmpNetwork object to a station.*

This brings up the **`New`** dialog box. Type a name in the **`name`** field (or accept the default name) and select `enable` to enable the network.

Step 4 Click **`OK`** to add the SnmpNetwork to the station.

The SnmpNetwork appears in the Driver Manager view with the name that you assigned in the previous step. The **`Status`** should be "`{ok}`" and **`Enabled`** field set to "`true`." The SnmpNetwork node should also appear under your Drivers node in the Workbench nav side bar.

Note: *You must save and restart the station before the SnmpNetwork object will start network communications.*

Designing an Snmp network application

There are two types of Snmp applications, both of which are represented as NiagaraAX "devices". Depending on how you plan to use the Niagara Snmp integration, you need to set up at least one or possibly both of these applications. Use the appropriate procedure or procedures to design the Snmp network, as described below:

- Snmp Local Device (Agent)
  Set up and use this NiagaraAX device for creating a "virtual" Snmp agent that can expose data from outside Niagara to Snmp Managers. Every SnmpNetwork has a local device and only one SnmpAgent object may be added to an SnmpNetwork.
- Snmp Manager
  Set up and use this NiagaraAX device for creating a "client" type application that you use to manage one or more Snmp Agent devices on your network. Under the Snmp Manager, SnmpDevice objects represent remote SnmpAgent devices that are configured to expose data from an actual Snmp device. You can add many SnmpDevice objects under your SnmpNetwork.

To set up an Snmp local device (agent application)

When you add an SnmpNetwork object, as described in "To add an SnmpNetwork to the station" on page 2-1, a Local Device component comes with it.

To set up an Snmp local device, do the following:

Step 1 Right-click on the SnmpNetwork node and select **Views > Property Sheet** to display the SnmpNetwork property sheet in the view pane.

Step 2 The Enabled property should be set to `True`. If not, select True from the option list.

Step 3 Set the **Snmp Receive Requests** property to `Enabled (can send traps)`. This allows the local device to receive outside request messages from external SNMP sources and to send trap messages.

Step 4 Click the property sheet **Save** button.

You are now ready to add Snmp Agent Points under the local device in order to expose Niagara data to SNMP requests from outside sources.

Refer to "Create Snmp proxy points" on page 2-3 for details about adding Snmp Agent Points.

To set up an Snmp manager application

For SNMP manager applications, you add SnmpDevice objects to your SnmpNetwork to represent actual SNMP agent devices that you want to manage.

To set up an Snmp manager application, do the following:

Step 1 Under the Drivers node of your station, double click on the SnmpNetwork node in the nav side bar. The Snmp Device manager view displays.

Step 2 Using the palette side bar controls, open the local SNMP module palette, expand the **Client** folder and copy-and-paste (or drag and drop) an SnmpDevice object into the view pane. The **Name** dialog box appears.

Step 3 Name the SnmpDevice, as desired and click the **OK** button. The device is added to the Snmp Device Manager view and represents *one* SNMP agent device on your network. You can add more of these objects, if needed, each representing a single SNMP agent device under your Snmp manager application.

Step 4 Double click on the added device in the Snmp Device Manager view. The **Edit** dialog appears.

Step 5 In the **Edit** dialog box, set the Snmp Version and Ip Address (actual device Ip address) fields, then click the OK button. The SnmpDevice is added under the SnmpNetwork. Refer to "SNMP device configuration" on page 3-6 for more complete Snmp device configuration information.

Create a MIB

You need to have a custom MIB for any Local Device component that contains objects that you want to expose to an SNMP Network Manager. If you already have a MIB that suits your needs, then you may not need to create a new one. You can create your own custom MIB for a "local device" (▣) then edit it, if desired, and save it to any read-write accessible location on your host. This custom MIB includes entries for the current MIB objects that are contained in your station.

Create a MIB

This procedure describes how to use the CreateMIB feature on a Snmp Local Device. The procedure requires that you have established a NiagaraAX workbench connection to the desired SNMP-licensed station with SnmpNetwork properly configured. For more information, refer to "About locally generated MIBs" on page 3-17.

Step 1 In the workbench nav tree, expand your station tree to expose the Local Device (**Station > Config > Drivers > SnmpNetwork**).

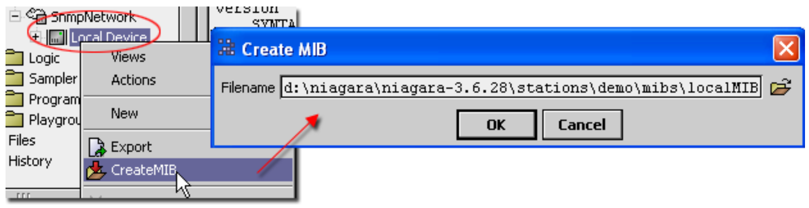
The Local Device displays under the SnmpNetwork node

.

Step 2 Right-click on the Local Device node and select **CreateMIB** from the popup menu.

The **Create MIB** dialog box displays, as shown below.

Figure 2-1 *Create and save a custom MIB file*



Filename d:\niagara\niagara-3.6.28\stations\demo\mibs\localMIB

Step 3 In the **Create MIB** dialog box, type the path or browse to a location to specify where you want to save the MIB file and click the **OK** button.

The MIB file displays in the MIB editor.

Step 4 If you need to make changes to the MIB file, use the editor to *carefully* edit the text and click the **OK** button when you are finished.

The file is saved and the MIB editor closes. The custom MIB file is now available for using with "Discovery" and value updates ("Walking the MIB").

Create Snmp proxy points

As with device objects in other drivers, each SnmpDevice has a Points extension that serves as the container for proxy points. In an Snmp Network, the default view for the Points extension is the **Snmp Point Manager** for a Manager application or the **Snmp Agent Point Manager** for an Agent (Local Device) application. You use this Point Manager view to add Snmp proxy points under any SnmpDevice.

For general information about using the Point Manager view, refer to "About the Point Manager" in the *NiagaraAX-3.x Drivers Guide.*

Note: *The* **Snmp Agent Point Manager** *works differently than Snmp Point Manager. For example, Snmp Agent Point Manager does not have a* **Discover** *button for adding proxy points. Typically, you add new points by copying Agent Points from the SNMP palette. In a Manager (Client) Snmp application, candidates for Snmp proxy points are always determined by using the Discover button and MIB files to find valid points and values on an Snmp Agent on the network. Once you discover these points, this information is then known to Niagara, and can be added using the* **Add** *or* **Match** *buttons.*

The following procedures describe how to add proxy points:

- To create Snmp Agent proxy points
- To create Snmp Client proxy points
- To create Snmp Client proxy points manually

To create Snmp Agent proxy points

Snmp Agent proxy points are typical point types with a special Snmp Agent extension included. To add Snmp *Client* proxy points, refer to "To create Snmp Client proxy points" on page 2-4.

To create Snmp Agent proxy points in a device, you must first have a properly configured SnmpNetwork with a Local Device under it.

Step 1 In the nav side bar, under the station **Snmp Network** node, expand the **Local Device** node and double-click on the **Points** node.

The **Snmp Agent Point Manager** displays.

Step 2 In the **Snmp Agent Point Manager**, click the **New** button at the bottom of the view (or right-click in the view and select **New** from the popup menu). The **New** dialog box appears.

Step 3 In the **New** dialog box, do the following:

- in the **Type to Add** field, select the type of proxy point that you want to add from the option list.
- in the **Number to Add** field, type in a number to indicate the quantity of proxy points that you want to add.
- click the **OK** button.

Another **New** dialog box appears.

Step 4 In the **New** dialog box, you can edit proxy point properties before each point is added in the Niagara station. OID values are automatically assigned for each point.

Note the following about entries in the **Add** dialog box:

- **Name**
  This property is the unique point name. This is the Niagara point name only—change if needed (does *not* affect the actual SNMP node).
- **Type**
  It is a Niagara control point type to use for the proxy point.
  Note: *Unlike other editable entries in the Add dialog, you cannot edit* **Type** *later.*
- **Index**
  As SnmpAgent proxy points are added to the SnmpAgent device, they are automatically assigned an index value that corresponds to the column index of the object created for it in the Tridium Input or Output Table.
- **Default Value**
  The default value is used for the output of the proxy point on startup prior to being set (by an external SNMP SET request). You can manually reset the proxy point output to the default value at any time by selecting the **Action > Reset Point To Default** action.
- **Enabled**
  This property allows you to set the proxy point in service (with a `true` value) or to set it out of service (with a `false` value).
- **Device Facets**
  This property represents the device proxy point facets that affect how the value should be displayed in Niagara.
- **Facets**
  This property represents the parent Niagara proxy point's facets, that affect how the value should be displayed in Niagara.
- **Conversion**
  This property specifies the conversion to use between the "read value" (in Device Facets) and the parent point facets, where "Default" is typically used.
- **Tuning Policy Name**
  This property specifies the Snmp service type to use when binding to this item.

Step 5 When you have Snmp proxy point(s) configured properly for your usage, click **OK**.

The proxy points are added to the station, and appear listed in the Snmp Manager view as well as under the **Points** node in the nav side bar.

- If online with the SnmpNetwork, points will poll for current values.
- If programming offline, all proxy points appear down (yellow).

To create Snmp Client proxy points

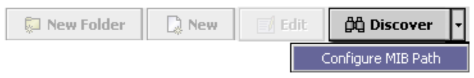
Snmp Client proxy points are typical point types with a special Snmp Client extension included. To add Snmp *Agent* proxy points, refer to "To create Snmp Agent proxy points" on page 2-3.

To create Snmp Client proxy points in a device, you must first have a properly configured SnmpNetwork with an SnmpDevice under it and a valid MIB file that is accessible from your workbench application.

Step 1 In the nav side bar, under the station **Snmp Network** node, expand the **Snmp Device** node that you are using and double-click on the **Points** node. The **Snmp Point Manager** displays.

Step 2 If you need to use Dependent MIBs, click the **Discover** button's drop-down arrow to open the **Specify Dependent MIB Directories** dialog box where you can specify one or more directories that hold your dependent MIBs.

Figure 2-2 *Adding dependent MIBs*



Step 3 In the **Snmp Point Manager**, click the **Discover** button at the bottom of the view. The **Select MIB(s) to load** dialog box appears.

Step 4 In the **Select MIB(s) to load** dialog box, click the file chooser button to find and select the appropriate MIB file from the **File Chooser** dialog box, then click the **Open** button to load the MIB.

The "MIB to load" field is populated with the selected MIB.

Step 5 In the **`Select MIB(s) to load`** dialog box, choose the `Walk the MIB?` option, if desired, and click the **`OK`** button.

The Snmp Point Manager view displays the discovered points in a table at the top of the view pane. If you chose the `Walk the MIB?` option in the previous step, any discovered values appear in the Values column of the table. Also, the **`Job Status Bar`**, at the top of the view, indicates the progress and completion of the MIB-walk job you just initiated.

Step 6 In the **`Discovered`** pane, select one or more of the discovered points that you want to add as proxy points and click the **`Add`** button. The **`Add`** dialog box appears.

In the **`Add`** dialog box, you can edit proxy point properties before each point is added in the Niagara station. OID values are automatically assigned for each point. Refer to "About Point Discover, Add and Match (Learn Process)" in the *NiagaraAX-3.x Drivers Guide* for more details about using the Add dialog box.

Note the following about entries in the **`Add`** dialog box:

- **Name**
  This property is the unique point name. This is the Niagara point name only—change if needed (does *not* affect the actual SNMP node).
- **Type**
  It is a Niagara control point type to use for the proxy point.
  Note: *Unlike other editable entries in the Edit dialog, you cannot edit* Type *later.*
- **Object Identifier**
  As SnmpAgent proxy points are added to the SnmpAgent device, they are automatically assigned an index value that corresponds to the column index of the object created for it in the Tridium Input or Output Table.
- **Variable Type**
  This property specifies the value type that is written out to the SNMP device at the location of the OID specified in the **`Object Identifier`** property. This property is usually automatically set when the Snmp proxy point is automatically created but must be typed in when creating the proxy point manually.
- **Enabled**
  This property allows you to set the proxy point in service (with a `true` value) or to set it out of service (with a `false` value).
- **Device Facets**
  This property represents the device proxy point facets for how the value should be displayed in Niagara.
- **Facets**
  This property represents the parent Niagara proxy point's facets, for how the value should be displayed in Niagara.
- **Conversion**
  This property specifies the conversion to use between the "read value" (in Device Facets) and the parent point facets, where "Default" is typically used.
- **Tuning Policy Name**
  This property specifies the Snmp service type to use when binding to this item.

Step 7 When you have the proxy point(s) configured properly for your usage, click **`OK`**.

The points are added to the station, and appear listed in the **`Database`** pane of the Snmp Point Manager view.

- If online with the SnmpNetwork, points will poll for current values.
- If programming offline, all proxy points appear down (yellow).

To create Snmp Client proxy points manually

Snmp Client proxy points are typical point types with a special Snmp Client extension included. To add Snmp *Agent* proxy points, refer to "To create Snmp Agent proxy points" on page 2-3.

To manually create Snmp Client proxy points in a device, do the following:

Step 1 In the nav side bar, under the station **`Snmp Network`** node, expand the **`Snmp Device`** node that you are using and double-click on the **`Points`** node. The **`Snmp Point Manager`** displays.

Step 2 In the **`Snmp Point Manager`**, click the **`New`** button at the bottom of the view. The **`New`** dialog box appears.

Step 3 In the **`New`** dialog box:

- in the **Type to Add** field, select the type of proxy point that you want to add from the option list.
- in the **Number to Add** field, type in a number to indicate the quantity of proxy points that you want to add.
- click the **OK** button.

A second **New** dialog box appears.

Step 4 In the **New** dialog box, you can edit proxy point properties before each point is added in the Niagara station. OID values are NOT automatically assigned for the added point because the point has not been "discovered".

Note the following about entries in the **Add** dialog box:

- **Name**
  This property is the unique point name. This is the Niagara point name only—change if needed (does *not* affect the actual SNMP node).
- **Type**
  It is a Niagara control point type to use for the proxy point.
  Note: *Unlike other editable entries in the Add dialog, you cannot edit* **Type** *later.*
- **Object Identifier**
  Because the Snmp proxy points are not "discovered" when you use the **New** button, the OID values must be typed in manually.
- **Variable Type**
  This property specifies the type of the value written out to the SNMP device at the location of the OID specified in the **Object Identifier** property. This property is usually automatically set when the Snmp proxy point is automatically created but must be typed in when creating the proxy point manually.
- **Enabled**
  This property allows you to set the proxy point in service (with a `true` value) or to set it out of service (with a `false` value).
- **Device Facets**
  This property represents the device proxy point facets that affect how the value should be displayed in Niagara.
- **Facets**
  This property represents the parent Niagara proxy point's facets that affect how the value should be displayed in Niagara.
- Conversion
  This property specifies the conversion to use between the "read value" (in Device Facets) and the parent point facets, where "Default" is typically used.
- **Tuning Policy Name**
  This property specifies the Snmp service type to use when binding to this item.

Step 5 When you have the proxy point(s) configured properly for your usage, click **OK**.

The points are added to the station, and appear listed in the **Database** pane of the Snmp Point Manager view.

- If online with the SnmpNetwork, points will poll for current values.
- If programming offline, all proxy points appear down (yellow).

CHAPTER 3

Niagara SNMP Concepts

This section describes the SNMP concepts that comprise NiagaraAX SNMP.

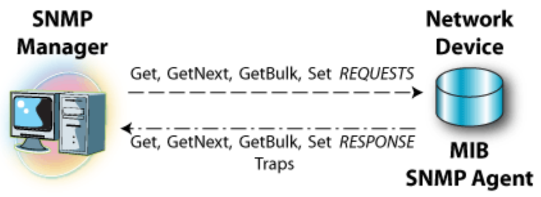
The following main sections are included:

About SNMP network architecture

Simple Network Management Protocol (SNMP) is a network-management protocol used almost exclusively in TCP/IP networks. SNMP provides a means to monitor and control network devices, and to manage configurations, statistics collection, performance and security on a network. SNMP uses a distributed architecture consisting of entities called “managers” and “agents”.

The SNMP agent exchanges network management information with SNMP manager software running on a network management system (NMS), or host. The agent responds to requests for information and actions from the manager. The agent also controls access to the agent's Management Information Base (MIB), the collection of objects that can be viewed or changed by the SNMP manager (see Figure 3-1).

Figure 3-1 *SNMP communication*


SNMP Manager
Network Device
Get, GetNext, GetBulk, Set REQUESTS
Get, GetNext, GetBulk, Set RESPONSE
Traps
MIB
SNMP Agent

Communication between the agent and the manager occurs in one of the following forms:

- **Get, GetBulk, and GetNext requests**
  The manager requests information from the agent; the agent returns the information in a Get response message.
- **Set requests**
  The manager changes the value of a MIB object controlled by the agent; the agent indicates status in a Set response message.
- **Traps notification**
  The agent sends traps to notify the manager of significant events that occur on the network device.

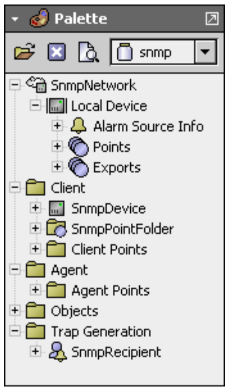
In Niagara, the SNMP driver uses the standard NiagaraAX network architecture. See "About Network architecture" in the *NiagaraAX-3.x Drivers Guide* for more details. The SNMP driver provides the components necessary to integrate SNMP devices and data into the Niagara environment. This driver currently supports SNMP versions 1 and 2. The SNMP Driver can be used to set up and serve both the manager and the agent applications:

- **SNMP Manager**
  A manager is an SNMP application that generates queries to SNMP agent applications and receives traps from SNMP agent applications.
  The Niagara SNMP Driver can be set up as a "Manager" (or "Client") to monitor a network of SNMP devices. This configuration includes a view for compiling MIB files to determine the data that is contained within an SNMP device. Once "discovered", the data points can be set up to monitor and control the SNMP devices. The driver can also handle receiving and processing unsolicited SNMP Trap Messages.
- **SNMP Agent**
  An agent is an SNMP application that responds to queries from SNMP manager applications. The SNMP agent is responsible for retrieving and updating local management information based on the requests of the SNMP manager. The agent also notifies registered managers when significant events or traps occur.
  The Niagara SNMP Driver can be set up to act as an SNMP agent on a network to *serve* information to an outside SNMP manager. The SNMP Agent is typically configured to expose Niagara data to the SNMP data requests and can also generate SNMP v1 trap messages based on Niagara alarms. Every SnmpNetwork component has a "Local Device" component. The local device is a frozen slot on the SnmpNetwork component that can be "disabled" but cannot be removed.

About SNMP palette components

Open the SNMP palette in Workbench to view components used in an SnmpNetwork.

Figure 3-2 *SNMP palette*



Following is a summary of the palette contents:

- **SnmpNetwork**
  This component contains the required Local Device component and its child components:
  - Alarm Source Info component
  - Points component.
  - Exports component
- **Client components**
  This container holds the components that are typically used for setting up a client SNMP (Manager) network, including the SnmpDevice component and its child components:
  - Alarm Source Info component
  - Points component
  - Traps component

  Additionally, a specially marked SnmpPointFolder is provided for organizing points, if desired. The standard point types are located in the Client Points folder as well. Most of the time you will probably use the Point Manager View to add points under the client device.
- **Agent components**
  The agent components include Agent Points that are standard point types with proxy extensions pre-configured for an SNMP device.

- **Objects**
  This folder contains a set of proxy points that can be exposed as individual points instead of the table presentation used for agent proxies and export points. The points allow values to be exposed as any supported SNMP ASN data type. To use these, add the SnmpObjectDeviceExt (from this folder in the palette) to the LocalDevice.
- **Trap generation**
  This folder contains the SnmpRecipient component.

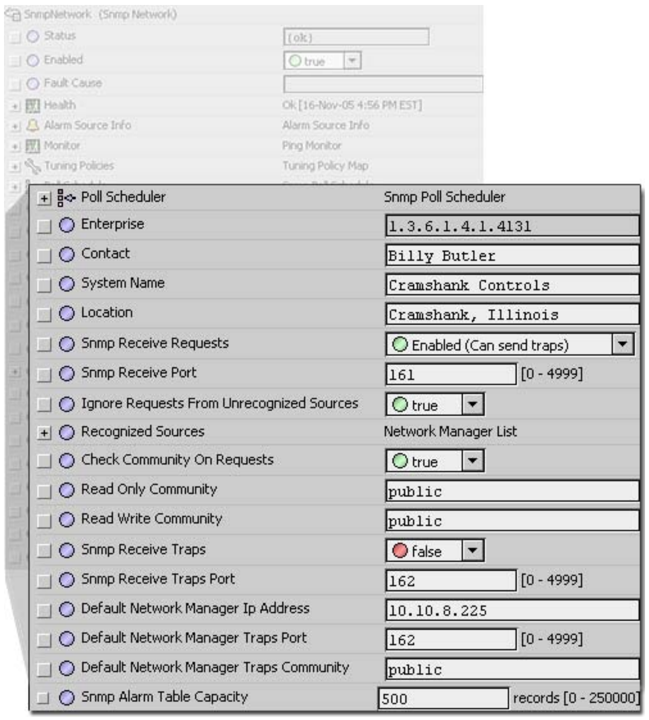
SNMP network configuration

SNMP devices communicate using the SNMP protocol over a network. The SnmpNetwork is designed to handle multiple SnmpDevices using the SNMP version 1 or SNMPv2 protocol (the protocol version can be selected by the user for each SnmpDevice independently). Configuration of the SnmpNetwork object is done through changing its properties to fit the SNMP integration. Only one SnmpNetwork object can be placed in a station.

SnmpNetwork component properties

By selecting the Property Sheet view of an SnmpNetwork object, you can view and configure the following SNMP properties that apply to the entire network:

Figure 3-3 *SNMP network properties*



SnmpNetwork (Snmp Network)

Status	{ok}
Enabled	true
Fault Cause	
Health	Ok [16-Nov-05 4:56 PM EST]
Alarm Source Info	Alarm Source Info
Monitor	Ping Monitor
Tuning Policies	Tuning Policy Map
Poll Scheduler	Snmp Poll Scheduler
Enterprise	1.3.6.1.4.1.4131
Contact	Billy Butler
System Name	Cramshank Controls
Location	Cramshank, Illinois
Snmp Receive Requests	Enabled (Can send traps)
Snmp Receive Port	161 [0 - 4999]
Ignore Requests From Unrecognized Sources	true
Recognized Sources	Network Manager List
Check Community On Requests	true
Read Only Community	public
Read Write Community	public
Snmp Receive Traps	false
Snmp Receive Traps Port	162 [0 - 4999]
Default Network Manager Ip Address	10.10.8.225
Default Network Manager Traps Port	162 [0 - 4999]
Default Network Manager Traps Community	public
Snmp Alarm Table Capacity	500 records [0 - 250000]

A description of each of the SNMP specific properties follows. For more details on properties specific to all types of network objects, refer to the *NiagaraAX-3.x Drivers Guide.*

- **Poll Scheduler**
  Used to configure device-level polling. See the *NiagaraAX-3.x Drivers Guide* for details on configuring the Poll Scheduler.
- **Enterprise**
  Displays the enterprise OID (Object Identifier) information for the station.
- **Contact**
  Specifies the system contact information for the station. This contact information is stored for the station and is read/write accessible via SNMP requests made to the station for OID 1.3.6.1.2.1.1.4.0.
- **System Name**
  Specifies the system name information for the station. This name information is stored for the station and is read/write accessible via SNMP requests made to the station for OID 1.3.6.1.2.1.1.5.0.

- **Location**
  Specifies the system location information for the station. This location information is stored for the station and is read/write accessible via SNMP requests made to the station for OID 1.3.6.1.2.1.1.6.0.
- **SNMP Receive Requests**
  Specifies whether the ability for SNMP request messages to be received by the station (only supports receiving GET, GETNEXT, or SET SNMPv1 or SNMPv2 messages) is enabled or disabled. When enabled, reception of SNMP request messages from external SNMP sources is possible (subject to the constraints placed on the reception of SNMP requests by the next six property fields). When disabled, reception of SNMP requests is not possible and any requests sent to it will be dropped.

Caution *When the ability to receive SNMP request messages (or SNMP trap messages) is enabled, your station may become insecure and subject to unauthorized SNMP requests from other sources. The additional properties for limiting the SNMP requests processed (by restricting the source IPs that may send SNMP requests and adding the ability to change the community string fields for read/write access) do not ensure the security of your station since no encryption is used. It is highly recommended that you only enable the reception of SNMP requests (or SNMP traps) when your station is on a secure network where SNMP requests from unauthorized sources are restricted.*

- **SNMP Receive Port**
  Specifies the port that the station uses to receive SNMP requests from external SNMP sources (such as a network manager). The default is port 161, the standard SNMP port.
- **Ignore Requests From Unrecognized Sources**
  Specifies whether to enable or disable the ability for SNMP request messages to be received from only recognized sources (specified in the 'Recognized Sources' property). When enabled, a received SNMP request message is first checked for its source IP, and if this source IP matches any one of the source IP addresses specified in the 'Recognized Sources' field, the request is processed. If the source IP does not match, then the request is disregarded. When this property is disabled, SNMP requests from any source IP will be processed.
- **Recognized Sources**
  This component can be configured with source IP addresses to specify a list of recognized network managers. This list is used if the 'SNMP Receive Requests' and 'Only Accept Requests From Recognized Sources' properties are both enabled. It contains a list of source IP addresses that will be searched whenever an incoming SNMP request is received, and if the source of that request matches a source IP in this list, then the request will be processed. Otherwise, the request will be dropped. Useful for security purposes to ensure that the station only responds to known sources. This component has two actions used to configure the source IP address list:
  - Add Address
    Prompts the user to enter a source IP address to add to the list. As source IP are added, they appear in the list as 'Source_IP_Address' entries under the component. These entries are editable.
  - Clear All Addresses
    Clears the list of source IP addresses. All 'Source_IP_Address' entries will be removed.
- **Check Community On Requests**
  Specifies whether to enable or disable checking the community string field on a received SNMP request message before processing the request. When enabled, a received SNMP request message is first checked for its community string, and if this community string matches the community string specified in the 'Read Only Community' field (for GET or GETNEXT requests) or the 'Read Write Community' field (for GET, GETNEXT, or SET requests), the request is processed. If the community string does not match for the appropriate read/write access, then the request is disregarded. When this property is disabled, SNMP requests with any community string field will be processed.
- **Read Only Community**
  Only valid if the 'Check Community On Requests' property is enabled, this property specifies the community string field that incoming SNMP request messages must contain in order to process a read-only request (GET or GETNEXT request). The default value is “public”.
- **Read Write Community**
  Only valid if the 'Check Community On Requests' property is enabled, this property specifies the community string field that incoming SNMP request messages must contain in order to process a read-write request (GET, GETNEXT, or SET request). The default value is “public”. NOTE: This property has priority over the 'Read Only Community' property.

- **SNMP Receive Traps**
  Specifies whether to enable or disable the ability for SNMP trap messages to be received by the station. When enabled, reception of SNMP trap messages from external SNMP devices is enabled, and any received trap messages will be routed to the Alarm Class specified by the 'Alarm Class For Received Traps' property, subject to the constraints of the 'Only Process Recognized Traps' property.

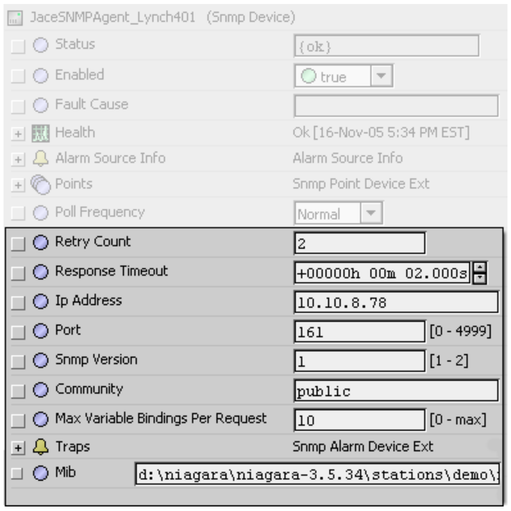
Caution *When the ability to receive SNMP request messages (or SNMP trap messages) is enabled, your station may become insecure and subject to unauthorized SNMP requests from other sources. The additional properties for limiting the SNMP requests processed (by restricting the source IPs that may send SNMP requests and adding the ability to change the community string fields for read/write access) do not ensure the security of your station since no encryption is used. It is highly recommended that you only enable the reception of SNMP requests (or SNMP traps) when your station is on a secure network where SNMP requests from unauthorized sources are restricted.*

- **SNMP Receive Traps Port**
  Specifies the port that the station uses to receive SNMP trap messages from external SNMP devices. The default is port 162, the standard SNMP traps port.
- **Default Network Manager Ip Address**
  Specifies the default IP address of the network manager to use for reporting information. Any SnmpRecipients configured to use the SnmpNetwork's default network manager will report SNMP trap messages generated by the station to the host at this IP address (see SnmpRecipient Configuration). This default network manager IP will also get sent a 'Cold Start' trap message from the Niagara station when the station first starts.
- **Default Network Manager Traps Port**
  Specifies the port to use for outgoing SNMP trap messages sent to the default network manager (i.e. the port on the default network manager where SNMP trap messages are received). This field defaults to port 162 - the standard SNMP traps port.
- **Default Network Manager Traps Community**
  Specifies the community string field to use for outgoing SNMP trap messages (sent to the default network manager). The default value is “public”.
- **Snmp Alarm Table Capacity**
  This property allows you to specify the maximum size of the Snmp Alarm Table. The default size is 500 records. Using this property, you can disable the Snmp Alarm Table by setting the value to zero. The purpose of limiting the table size is to conserve memory. You can also choose to disable Snmp alarm storage by choosing to set the SnmpRecipient’s “Snmp Alarm Table” property to “Do not Store Received”.

SNMP device configuration

You place SnmpDevice objects under an SnmpNetwork to represent actual SNMP devices that you want to communicate with. Then you configure the SnmpDevice object by setting its properties to match the settings for the actual SNMP device. After device configuration, you can configure proxy points (at the device-level) for reading and writing actual data values on the SNMP device.

Figure 3-4 *SNMP device properties*



JaceSNMPAgent_Lynch401 (Snmp Device)

Status	{ok}
Enabled	true
Fault Cause	
Health	Ok [16-Nov-05 5:34 PM EST]
Alarm Source Info	Alarm Source Info
Points	Snmp Point Device Ext
Poll Frequency	Normal
Retry Count	2
Response Timeout	+00000h 00m 02.000s
Ip Address	10.10.8.78
Port	161 [0 - 4999]
Snmp Version	1 [1 - 2]
Community	public
Max Variable Bindings Per Request	10 [0 - max]
Traps	Snmp Alarm Device Ext
Mib	d:\niagara\niagara-3.5.34\stations\demo\

Note: *SnmpDevice objects may only exist under an SnmpNetwork object.*

You can view and edit the following SnmpDevice specific properties in the property sheet view of the SnmpDevice:

A description of the SNMP specific properties follows (displayed in the property sheet view). For more details on properties that are common to all types of device objects, please refer to the *NiagaraAX-3.x Drivers Guide* documentation.

- **Actions**
  The SnmpDevice object has one visible action:
  - `Ping`
    This action manually initiates a “ping” (check device status) on the actual SNMP device that the SnmpDevice object represents.
- **Retry Count**
  Specifies the number of times any individual SNMP request made to this SNMP Device will be re-tried when receiving a null response before considering the request to be a communication failure.
- **Response Time Out**
  Specifies the maximum amount of time to wait for a response after sending an SNMP request to this SNMP device. If no response is received in this amount of time after sending a request, the request will be considered failed, and the driver will either retry the request (per the 'Retry Count' property) or consider the transaction a failure.
- **Ip Address**
  Specifies the IP address of the actual SNMP device for which this SnmpDevice represents.
- **Port**
  Specifies the port to use for outgoing SNMP requests to the corresponding SNMP device for this SnmpDevice (i.e. the port on the external SNMP device where SNMP requests are received). The default is port 161 - the standard SNMP port.
- **SNMP Version**
  Specifies the version of the SNMP protocol to use for communication with this SNMP device. The default is 2 (representing SNMPv2), and the only other option is 1 (representing SNMPv1).
- **Community**
  Specifies the community string field to use for outgoing SNMP request messages sent to the SNMP device. The default value is “public”.
- **Max Variable Bindings Per Request**
  Specifies the maximum number of variable bindings to include in each SNMP request message sent to the SNMP device. The default value is 10, however, this value can be any integer value greater than or equal to 1. This is useful if the SNMP request messages sent during device-level polling become

too large (due to a large number of subscribed SNMP proxy points for the SnmpDevice), and you would like to split up the request messages into multiple requests (of shorter length). A value of 1 would cause individual requests for each subscribed SNMP proxy point's data value within the SnmpDevice on each poll cycle for the SnmpDevice.

- **Traps**
  This editable component stores a list of SNMP trap notification types created for this SnmpDevice using the MIBPointListManager. If the SnmpNetwork is configured to receive SNMP trap messages, any trap received from this SNMP device will first check to see if it has a corresponding trap notification type in this list in order to describe the meaning of the received trap. This component has one action:
  - Clear - Deletes the list of stored SNMP trap/notification types. All entries will be removed.

  For more information about the Trap device extension, including traps property descriptions, refer to "SNMP alarm device extension properties" on page 3-16.
- **Mib**
  This editable field allows you to specify the path to a MIB file. You can populate this field by typing directly in the field or by using the `Select MIB(s) to load` dialog box, which is available from the Snmp Point Manager view.

About SNMP proxy points

There are two basic categories of SNMP proxy points, as described below and in the following sections:

- **SNMP client (manager) proxy points**
  Client proxy points are used under an SNMP client (manager) application.
  Refer to "SNMP client proxy points" for information about these types of points.
- **SNMP agent (server) proxy points**
  Agent SNMP proxy points are used under an SNMP agent (local device) application.
  SnmpAgent proxy points are used to expose read-write and read-only Niagara data to an outside SNMP manager making requests to the station. A running station with an SnmpNetwork includes an Input and Output Table. These tables may contain one or more objects that can be exposed through GET, GETNEXT, or SET requests sent to it from an outside SNMP device (usually an outside SNMP manager). When SnmpAgent proxy points are added to an SnmpAgent under the SnmpNetwork, the Input Table (for read-write objects) or Output Table (for read-only objects) are populated and thus accessible via SNMP requests (supports both SNMPv1 and SNMPv2 GET, GETNEXT, or SET requests).
  To get a list of all of the agent data in the Input Table, you have to perform a series of GETNEXT and/or GET requests starting from the following root OID: 1.3.6.1.4.1.4131.1.4.
  Note: *The data contained in the Input Table will also accept SET requests to change the value of the data and the corresponding SnmpAgent proxy point.*

  To get a list of all of the agent data in the Output Table, perform a series of GETNEXT and/or GET requests starting from the following root OID: 1.3.6.1.4.1.4131.1.5.
  Note: *The data contained in the Output Table will not accept SET requests as it contains read-only data.*

  As SnmpAgent proxy points are added to the SnmpAgent device, they are automatically assigned an index value that corresponds to the column index of the object created for it in the Input or Output Table.
  Refer to "SNMP agent proxy points" for more information.
- **SNMP agent (server) export points**
  These types of points appear under the Snmp Export Table, which has the Snmp Export Manager as its default view. You can use the Snmp Export Manager view to discover and add control points from a Local Device to the Export Table. The discovery process uses a `Bql Query` dialog box to help you find and filter control points. Once these export points are in the Snmp Export Table they can be discovered by an SnmpNetwork manager using the Snmp Point Manager view.

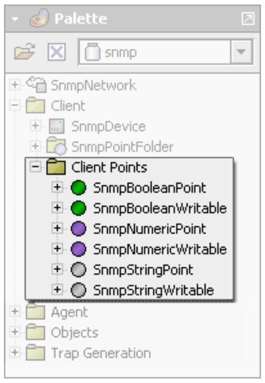
SNMP client proxy points

SNMP client proxy points must be located under the SNMP point device extension. SNMP proxy points are comprised of standard Niagara control points with unique SNMP proxy point extensions that define the SNMP-specific behavior of the point.

See the appropriate *NiagaraAX-3.x Drivers Guide* documentation for details on the basic control point types themselves.

You typically add SNMP client points under the SNMP points container during a "discovery" process using the SNMP Point Manager view. Use the **Add** or **Match** buttons to bring the points into the SNMP client application with the correct proxy point extension, based on your Type selection in the **Add** dialog box. Refer to "About the SNMP point manager view" on page 3-21 for more information about the Point Manager view. The client proxy points and extensions are also available in the **Client** folder of the SNMP palette, as shown in Figure 3-5.

Figure 3-5 *SNMP client proxy points*



Following is a list of the SNMP client proxy points:

- **SnmpBooleanPoint**
  This read-only control point type contains the SNMP Boolean proxy point extension.
- **SnmpBooleanWritable**
  This writable control point type contains the SNMP Boolean proxy point extension.
- **SnmpNumericPoint**
  This read-only control point type contains the Numeric SNMP proxy point extension.
- **SnmpNumericWritablePoint**
  This writable control point type contains the Numeric SNMP proxy point extension.
- **SnmpStringPoint**
  This read-only control point type contains the String SNMP proxy point extension.
- **SnmpStringWritable**
  This writable control point type contains the String SNMP proxy point extension.

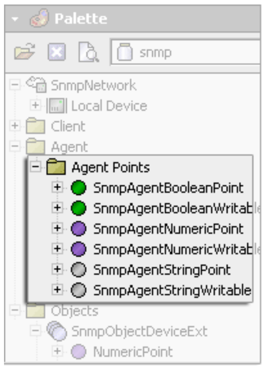
These SNMP proxy points support reading or writing the following SNMP variable types:

- Integer
- String (OCTET STRING)
- Object Identifier
- IpAddress
- Counter (both Counter32 and Counter64)
- Gauge
- TimeTicks

SNMP agent proxy points

SnmpAgent proxy points must be located under an SNMP Point Device Extension. SnmpAgent proxy points consist of standard Niagara control points with unique SnmpAgent Proxy Point Extensions which define the SnmpAgent-specific behavior of the point.

Figure 3-6 *SNMP agent proxy points*



The SnmpAgent proxy points leverage off the standard Niagara control point types. Refer to the appropriate *NiagaraAX-3.x Drivers Guide* documentation for details on the basic control point types.

Following is a list of the SNMP agent proxy points:

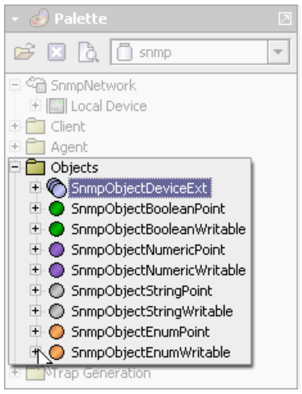
- **SnmpAgentBooleanPoint**
  This control point type hosts the following read-write SnmpAgent proxy point extension that populates the Input Table: SnmpAgentBooleanProxyExt.
- **SnmpAgentBooleanWritable**
  These control point types host the following read-only SnmpAgent proxy point extensions that populate the Output Table: SnmpAgentBooleanProxyExt.
- **SnmpAgentNumericPoint**
  This control point type hosts the following read-write SnmpAgent proxy point extension that populates the Input Table: SnmpAgentNumericProxyExt
- **SnmpAgentNumericWritablePoint**
  These control point types host the following read-only SnmpAgent proxy point extensions that populate the Output Table: SnmpAgentNumericWritableProxyExt.
- **SnmpAgentStringPoint**
  This control point type hosts the following read-write SnmpAgent proxy point extension that populates the Input Table: SnmpAgentStringProxyExt
- **SnmpAgentStringWritable**
  These control point types host the following read-only SnmpAgent proxy point extensions that populate the Output Table: SnmpAgentStringWritableProxyExt.

SNMP object proxy points

SnmpAgent proxy points must be located under an SNMP Point Device Extension. SnmpAgent proxy points consist of standard Niagara control points with unique SnmpAgent Proxy Point Extensions which define the SnmpAgent-specific behavior of the point.

(see the appropriate Niagara documentation for details on the control point types themselves).

Figure 3-7 *SNMP object proxy points*



The SnmpAgent proxy points leverage off the standard Niagara control point types. Refer to the *NiagaraAX-3.x Drivers Guide* documentation for details on the basic control point types.

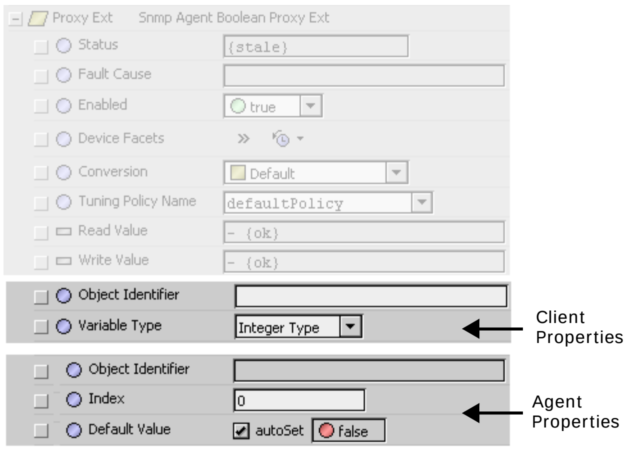
Following is a list of the SNMP object proxy points:

- SnmpObjectDeviceExt
  This component is a container for any of the set of SnmpObject types, listed below. The default view of this extension is the Snmp Object Manager view.
- SnmpObjectBooleanPoint
- SnmpObjectBooleanWritable
- SnmpObjectNumericPoint
- SnmpObjectNumericWritablePoint
- SnmpObjectStringPoint
- SnmpObjectStringWritable
- SnmpObjectStringWritable
- SnmpObjectStringWritable

SNMP proxy point extensions

Each SNMP proxy point type contains a corresponding proxy point extension. SNMP proxy point extensions have the same core properties as other proxy point extensions (described in the *NiagaraAX-3.x Drivers Guide*), plus some SNMP-specific properties, which are described here. These extension properties are similar for both client and agent proxy point application. There are some differences that are noted in the following list of property descriptions:

Figure 3-8 *SNMP proxy point extension properties*


Proxy Ext Snmp Agent Boolean Proxy Ext
Status {stale}
Fault Cause
Enabled true
Device Facets
Conversion Default
Tuning Policy Name defaultPolicy
Read Value - {ok}
Write Value - {ok}
Object Identifier
Variable Type Integer Type
Object Identifier
Index 0
Default Value autoSet false

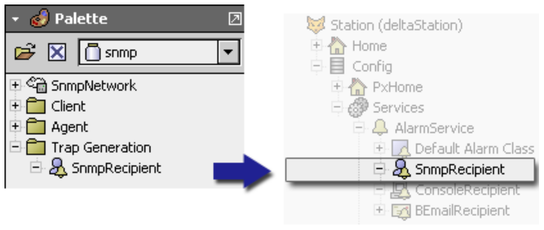
- **Object Identifier (Client)**
  Specifies the OID of the actual SNMP device where the data is to be read from or written to.
- **Object Identifier (Agent)**
  Displays the full OID to use for accessing the data value for this point from the Input Table (or Output Table) by an outside SNMP manager. SNMP GET or SET requests would use this full OID to access the value of the proxy point.
- **Variable Type**
  This property specifies the variable type of the value that is written out to the SNMP device at the location of the OID specified in the 'Object Identifier' property.
  Note: *This property is usually automatically set when the SNMP proxy point is created from the SnmpPointManager (as long as the type field for the MIB entry created matches one of the known types). Otherwise the user will have to manually set this property for the appropriate variable type before input values are properly written to this point in the actual SNMP device.*
- **Index (Agent)**
  The column index in the Input Table (or Output Table) used to locate the value of the proxy point. It is simply the last number in the 'Object Identifier' (OID) value. For example, if the index has a value of 1, then the OID containing the value of the point is 1.3.6.1.4.1.4131.1.4.1.3.1. If the index is 2, this corresponds to an OID of 1.3.6.1.4.1.4131.1.4.1.3.2, and so on.
- **Default Value (Agent)**
  This property is the default value used for the output of the proxy point on startup prior to being set (by an external SNMP SET request). If 'autoSet' is checked, then the default value is automatically set (changed) whenever a new external SNMP SET request is received. If 'autoSet' is unchecked, then the Niagara user can specify a permanent default value (any external SNMP SET request will not change the default value). The Niagara user can manually reset the proxy point's output to the default value at any time by selecting the 'Reset Point To Default' action.

SnmpRecipient configuration

SnmpRecipient components extend from standard Niagara AlarmRecipients. They function the same as the standard AlarmRecipient component (See “Types of alarm recipients” on page 35.in the *NiagaraAX-3.x User Guide*) except that the SnmpRecipient has added properties for configuring and handling the SNMP specific behavior. In order to use the SnmpRecipient, you must place the SnmpRecipient component under the station Alarm Service, as shown in Figure 3-9.

Note: *A configured SnmpNetwork object must exist in the station in order for the SnmpRecipient to function properly.*

Figure 3-9 *SnmpRecipient under the AlarmService*



Notice that the **`Alarm`** slot of the Alarm Class component is linked to the **`Route Alarm`** slot of the SnmpRecipient component (see Figure 3-11). This ensures that Niagara alarms are routed to the SnmpRecipient appropriately. Be sure to use this connection for each Alarm Class-to-SnmpRecipient link in your station.

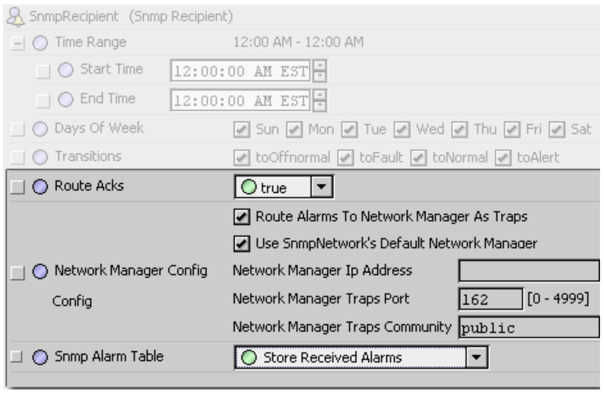
SnmpRecipient-specific properties are described in the “SnmpRecipient properties” on page 3-11. Configure the SnmpRecipient component to generate and receive SNMP traps as described in the following sections:

- Generating SNMP traps
  Use the SnmpRecipient component to generate and send Niagara alarms (in the form of SNMPv1 traps) to a defined SNMP manager whenever a new Niagara alarm record is routed to the SnmpRecipient.
- Receiving SNMP traps
  Use the SnmpRecipient to store any received Niagara alarm records in the Alarm Table of the SnmpNetwork. The SnmpRecipient receives the alarm record from the linked Alarm Class and packages it into an SNMPv1 trap message to send to the network manager.

SnmpRecipient properties

By selecting the Property Sheet view of an SnmpRecipient component, you can view and configure the SnmpRecipient properties, as shown in Figure 3-10.

Figure 3-10 *SnmpRecipient properties*



SnmpRecipient (Snmp Recipient)

Time Range	12:00 AM - 12:00 AM
Start Time	12:00:00 AM EST
End Time	12:00:00 AM EST
Days Of Week	✓ Sun ✓ Mon ✓ Tue ✓ Wed ✓ Thu ✓ Fri ✓ Sat
Transitions	✓ toOffnormal ✓ toFault ✓ toNormal ✓ toAlert
Route Acks	true
	✓ Route Alarms To Network Manager As Traps
	✓ Use SnmpNetwork's Default Network Manager
Network Manager Config	Network Manager Ip Address
Config	Network Manager Traps Port: 162 [0 - 4999]
	Network Manager Traps Community: public
Snmp Alarm Table	Store Received Alarms

The Time Range, Days of Week, and Transitions properties are standard to all Alarm Recipients and detailed in the *NiagaraAX-3.x Drivers Guide*. The SNMP-specific properties are described in the following list:

- **Route Acks**
  Enable this property by selecting the `true` option. Trap acknowledgements are not routed if the `False` option is selected.
  - Route Alarms To Network Manager As Traps

When selected, this property option enables generating and sending SNMP version1 trap messages to the specified SNMP network manager whenever a new Niagara alarm record is received. It this option is not selected, then this service is not enabled and any received Niagara alarm record does not generate or send an SNMP version 1 trap message.

- Use SnmpNetwork's Default Network Manager
  When this property option is selected, Niagara alarms that are reported as traps, report using the network manager that is defined in the `Default Network Manager` properties. If this property option is not selected, then the alarms are reported as traps using the network manager that is defined in the `Network Manager Config` properties.

- **Network Manager Config**
  The following three properties are available only when the `Use SnmpNetwork's Default Network Manager` property option (check box) is *not* selected.
  - Network Manager Ip Address
    This property is a text field for defining the IP address of the network manager that is used for reporting the generated SNMP version 1 traps.
  - Network Manager Traps Port
    This property specifies the port to use for outgoing SNMP version 1 trap messages sent to the specified network manager (i.e. the port on the network manager where SNMP trap messages are received). This field defaults to port 162 - the standard SNMP traps port.
  - Network Manager Traps Community
    This property specifies the community string field to use for outgoing SNMP version 1 trap messages (sent to the network manager). The default value is "public".
- **Snmp Alarm Table**
  Select one of the following options from the option list:
  - Store Received Alarms
    This option specifies that alarms that are routed to this recipient are also cached in the Snmp Alarm Table. This is the default option.
  - Do Not Store Received Alarms
    This option disables the caching of alarms for the recipient.

About SNMP alarms (traps)

Niagara handles alarms using the station alarming service and its alarm classes. The SNMP driver uses these alarm classes in generating and receiving SNMP traps. Because alarm classes are standard Niagara objects, all Niagara alarms that generate trap messages (or all received SNMP trap messages that generate Niagara alarms) can be viewed in a Niagara alarm console. In order to generate and receive SNMP traps, you must properly configure the SnmpNetwork *and* SnmpRecipient components.

Refer to the following sections for more information about configuring the SNMP network for trap generation and trap reception:

- Generating SNMP traps
- Receiving SNMP traps

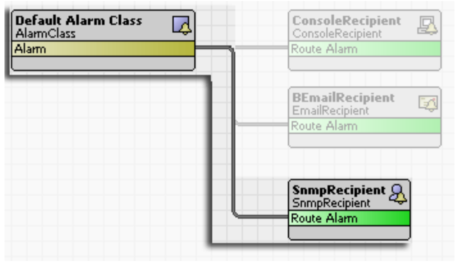
For additional information about Niagara alarming and the Niagara alarm console, refer to "About alarms" and "About the alarm console" in the *NiagaraAX-3.x User Guide*.

Generating SNMP traps

You can generate and send Niagara alarms (in the form of SNMPv1 traps) to a defined SNMP manager whenever a new Niagara alarm record is routed to the SnmpRecipient. In order to generate SNMP traps you must setup and configure the SnmpRecipient on a station running an SnmpNetwork. This includes placing an SnmpRecipient component under the AlarmService component, configuring it to enable trap generation and designating a valid SNMP network manager. Refer to "SnmpRecipient configuration" on page 3-11 for details.

Once the proper IP address and port are assigned for the SNMP network manager, the SnmpRecipient component must have a link to an appropriate Alarm Class under the Alarming Service in order to provide the Niagara Alarm source for the generation and routing of SNMP trap messages to the network manager. You add SnmpRecipient components under the AlarmService by copy-and-paste from the SNMP module palette. You can add as many SnmpRecipients as you need to configure your SNMP integration. Figure 3-11 shows an SnmpRecipient component linked to the Default Alarm Class. In this example, any Niagara alarms routed to the Default Alarm Class are also routed to the SnmpRecipient. The SnmpRecipient then packages the alarms into SNMP trap messages and sends them to the specified SNMP network manager.

Figure 3-11 *SnmpRecipient linked to default alarm class*



Note: *All SNMP traps generated and sent by the SnmpRecipient are SNMPv1 messages. The SNMP driver currently does not support generating and sending SNMPv2 trap messages.*

To configure a station for SNMP trap generation

To configure a station for SNMP trap generation, you must first have an SNMP network established and running in your station. (See "Adding an SnmpNetwork" on page 2-1).

Step 1 Open the SNMP palette in the Workbench palette side bar.

Step 2 Under the **`Trap Generation`** folder, drag and drop an **`SnmpRecipient`** component onto the **`AlarmService`** property sheet or onto the **`AlarmService`** component in the nav tree.

Step 3 Select the property sheet view of the SnmpRecipient and set the Time Range, Days of Week, and Transitions properties as desired. See "SnmpRecipient properties" on page 3-11 for details.

Step 4 On the SnmpRecipient property sheet, set the **`Route Acks`** property to `True` and choose the following options:

- `Route Alarms To Network Manager As Traps`
  If this option is *not* selected, traps are *not* routed.
- `Use SnmpNetwork's Default Network Manager`
  If this option is selected, traps report using the network manager that is defined in the **`Default Network Manager`** properties. If this property option is not selected, then the alarms are reported as traps using the network manager that is defined in the **`Network Manager Config`** properties.

Step 5 Set the following Network Manager Config properties if you choose to use an SnmpNetwork Manager other than the SnmpNetwork's Default Network Manager.

Note: *These properties are read-only unless you clear (do not select) the* `SnmpNetwork's Default Network Manager` *option field under the Rout Acks property.*

- `Network Manager IP Address`
  Type in the address of the SnmpNetwork manager that you want to use.
- `Use SnmpNetwork's Default Network Manager`
  If this option is selected, traps report using the network manager that is defined in the **`Default Network Manager`** properties. If this property option is not selected, then the alarms are reported as traps using the network manager that is defined in the **`Network Manager Config`** properties.
- `Network Manager Traps Community`
  This property provides a field for you to specify a string value to use for outgoing SNMP version 1 trap messages (sent to the network manager). The default value is "public".

Step 6 On the **`SnmpRecipient`** property sheet, choose one of the following options for the Snmp Alarm Table property.

- **Store Received Alarms**
  This option specifies that alarms that are routed to this recipient are also cached in the Snmp Alarm Table. This is the default option.
- **Do Not Store Received Alarms**
  This option disables the caching of alarms for the recipient.

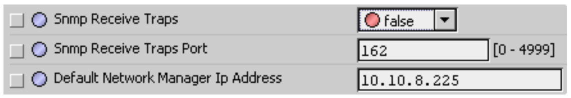
Step 7 On the **`SnmpRecipient`** property sheet, click the **`Save`** button. SNMP trap generation is configured and enabled.

Receiving SNMP traps

In order to receive SNMP traps you must configure the SnmpNetwork to receive traps. This includes setting the enabling property and specifying the port that listens for traps, as shown in Figure 3-12. You also need to configure properties that specify whether or not you want to receive traps from unrecognized sources. Refer to the following sections for more details about configuring SNMP components for receiving traps:

- "To configure a network manager (station) to receive SNMP traps" on page 3-14
- "SnmpRecipient properties" on page 3-11

Figure 3-12 *Configure SNMP network to receive traps*



Snmp Receive Traps	false	
Snmp Receive Traps Port	162	[0 - 4999]
Default Network Manager Ip Address	10.10.8.225	

Received trap messages are also routed to the SnmpDevice object and displayed in string form in the object's Last Trap Received property field.

To configure a network manager (station) to receive SNMP traps

To configure a station for receiving SNMP traps, you must first have an SNMP network established and running in your station (see "Adding an SnmpNetwork" on page 2-1). You also need to have an SnmpDevice component in your network application for any devices that may potentially send out trap messages (see "Designing an Snmp network application" on page 2-1).

To configure a network manager to receive SNMP traps, do the following:

Step 1 Select the property sheet view of the SnmpNetwork component.

Step 2 Set the **`SNMP Receive Traps`** property, to `True`.

Step 3 Set the **`SNMP Receive Traps Port`** property to the desired port number. (Port 162 is the default value).

Step 4 On the **`SnmpNetwork`** property sheet, click the **`Save`** button. SNMP traps that are directed to the SnmpNetwork manager are received and routed to the alarm class specified under the "Traps" device extension Alarm Class property of the client device.

Step 5 In the property sheet view of any potential trap-generating SNMP device, expand the Traps device extension component and set the following two properties, as desired:

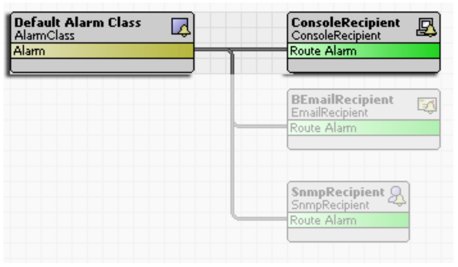
- **Alarm Class**
  Set this property to the alarm class that you want to use to route traps that are received from the parent device.
- **Ignore Unrecognized Traps**
  Set this property to True if you want to disregard trap messages (no alarm generation) that contain an unrecognized trap type. Set the property to False if you want to route all received traps to the designated alarm class, even if the trap type is not recognized.

Step 6 On the **`SnmpDevice`** property sheet, click the **`Save`** button. SNMP trap generation is configured and enabled.

Viewing Received Traps

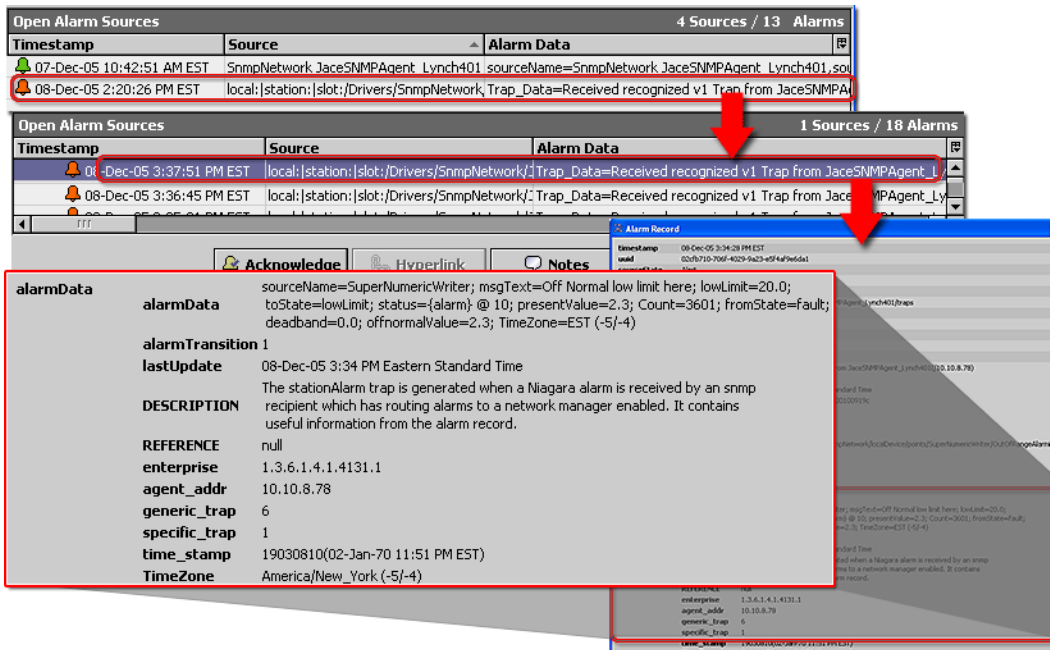
Traps information may be viewed in the alarm console. Route traps to the alarm console by linking the ConsoleRecipient to the appropriate AlarmClass component, as shown in Figure 3-13.

Figure 3-13 *ConsoleRecipient linked to default alarm class*



You can use the alarm console to view the trap data in the Alarm Data column. However, the Alarm Data column is very long and not totally viewable in the alarm console view. To read the Alarm Data information, double-click on the row that you want to read, and display the **`Open Alarm Sources`** view and then the **`Open Alarm Records`** dialog box, as shown in the Figure 3-14. The Alarm Record dialog box provides a full description of the message.

Figure 3-14 *Alarm console views*



Timestamp	Source	Alarm Data
07-Dec-05 10:42:51 AM EST	SnmpNetwork JaceSNMPAgent_Lynch401	sourceName=SnmpNetwork JaceSNMPAgent_Lynch401,sou
08-Dec-05 2:20:26 PM EST	local:|station:|slot:/Drivers/SnmpNetwork	Trap_Data=Received recognized v1 Trap from JaceSNMPA

Timestamp	Source	Alarm Data
08-Dec-05 3:37:51 PM EST	local:|station:|slot:/Drivers/SnmpNetwork/	Trap_Data=Received recognized v1 Trap from JaceSNMPAgent_L
08-Dec-05 3:36:45 PM EST	local:|station:|slot:/Drivers/SnmpNetwork/	Trap_Data=Received recognized v1 Trap from JaceSNMPAgent_Ly

alarmData	alarmData	sourceName=SuperNumericWriter; msgText=Off Normal low limit here; lowLimit=20.0; toState=lowLimit; status={alarm} @ 10; presentValue=2.3; Count=3601; fromState=fault; deadband=0.0; offnormalValue=2.3; TimeZone=EST (-5/-4)
	alarmTransition	1
	lastUpdate	08-Dec-05 3:34 PM Eastern Standard Time
	DESCRIPTION	The stationAlarm trap is generated when a Niagara alarm is received by an snmp recipient which has routing alarms to a network manager enabled. It contains useful information from the alarm record.
	REFERENCE	null
	enterprise	1.3.6.1.4.1.4131.1
	agent_addr	10.10.8.78
	generic_trap	6
	specific_trap	1
	time_stamp	19030810(02-Jan-70 11:51 PM EST)
	TimeZone	America/New_York (-5/-4)

For complete details about using the alarm console, including information about adding or removing alarm console columns, refer to "About the alarm console" in the *NiagaraAX-3.x User Guide*.

About SNMP alarmData

The Alarm Data field of the alarm console displays a received trap message using a `Trap_Data` facet in one of the following forms, depending on SNMP version:

Note: *The following typographical conventions apply to the examples:*

- Italicized values in angle brackets (< >) represent values that are taken from the station such as stored trap/notification type information and SnmpDevice IP addresses.
- Italicized values enclosed in brackets ([ ]) represent values that are taken from the received trap message itself.
- Variable bindings are displayed in the following form:
  [Variable Type]: [Value]
  Where 'Variable Type' is one of the following: OID, COUNTER, GAUGE, INT, IP ADDRESS, STRING, or TIMETICKS.
- For a null variable binding, the text is 'NULL OBJ '.

For a recognized SNMPv1 trap message from a known IP address:

```
"Received recognized v1 Trap from <SnmpDevice object name>(<IP Address>):
NAME= <Trap Name>;
VARIABLES=
<variable name>([variable binding taken from received trap]),
… … …;
DESCRIPTION= <Trap Description>;
REFERENCE= <Trap Reference>;
[DETAILS=
enterprise: [oid];
agent-addr: [ipaddress];
generic-trap: [integer];
specific-trap: [integer];
time-stamp: [timeticks]]"
```

For a recognized SNMPv2 trap message from a known IP address:

```
"Received recognized v2 Trap from <SnmpDevice object name>(<IP Address>):
NAME= <Trap Name>(<oid of trap>);
OBJECTS=
<variable name>([variable binding taken from received trap]),
… … …;
DESCRIPTION= <Trap Description>;
REFERENCE= <Trap Reference>"
```

SNMP alarm device extension properties

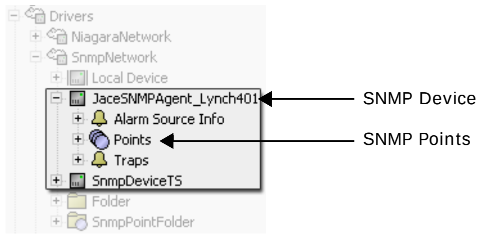
Common extension properties are described in the *NiagaraAX-3.x Drivers Guide* documentation. The following properties are SNMP-specific properties found under the SNMP Traps component:

- **Alarm Class**
  This property displays an option list that allows you to choose the desired local AlarmClass for routing traps that are received from this device.
- **Time Of Last Trap Received**
  Displays a timestamp of the last time (since station startup) that an SNMP trap message was received from the actual SNMP device represented by this SnmpDevice.
- **Last Trap Received**
  This StringElement output displays the detailed message of the last received SNMP trap message for this SnmpDevice since station startup.
- **Ignore Unrecognized Traps**
  This `True` or `False` option allows you to filter out any traps that are not recognizable based on the stored trap types for the source SnmpDevice. If this property is set to True, then the received unrecognized trap message will be disregarded (no alarms generated). If this property is disabled, then all received trap messages will be handled and routed to the specified Alarm Class whether they are recognizable or not.

About SNMP manager

You may configure the Niagara SnmpNetwork as an SNMP Manager to view and control points on a remote SnmpNetwork station. The remote station must contain an SNMP device component that is configured as an SNMP agent. The SNMP agent exposes data through the use of an SnmpAgent device and SnmpAgent proxy points (see "Configure the SnmpNetwork" on page 2-1 and "Create Snmp proxy points" on page 2-3). In this case alarms generated by the station can be stored in a Alarm Table (using an **SnmpRecipient** component) allowing the remote SNMP manager to view and acknowledge Niagara alarms via SNMP requests.

Figure 3-15 *SNMP manager devices*



"About locally generated MIBs" on page 3-17 describes the MIB file (`TridiumR3-MIB.my`) for a station running the SnmpNetwork. It also explains how it is used to view and control the station.

About MIBs

A Management Information Base (MIB) describes a database of objects that can be monitored by a network management system. SNMP uses standardized MIB formats that allow any SNMP tool to monitor any device defined by a MIB.

Each managed object in a MIB has a unique identifier that specifies things such as:

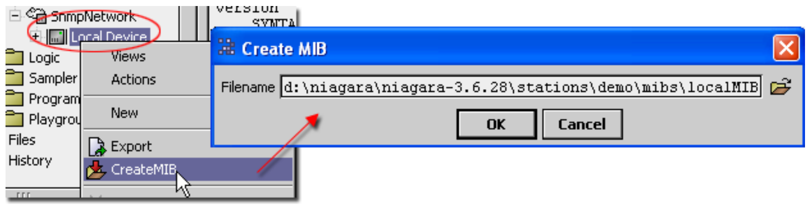
- object type (such as counter, string, or integer, for example)
- object access level (such as read or read/write)
- size restriction
- range information

The name space for MIB object identifiers is hierarchical. It is structured so that each manageable object can be assigned a globally unique name. Authority for parts of the name space is assigned to individual organizations. This allows organizations to assign names without consulting an Internet authority for each assignment. For example, the name space assigned to a (fictional) Corporation X might be 1.3.6.1.4.1.nnn (where n is an integer), which is defined in CORPX.MIB. Corporation X then has the authority to assign names to objects anywhere below that name space. The object identifier in the hierarchy is written as a sequence of sub identifiers beginning at the root and ending at the object with sub identifiers separated by a period.

About locally generated MIBs

You can create a custom MIB on a local device by selecting the **CreateMIB** menu item CreateMIB from the popup menu on an SnmpNetwork Local Device (see "Create a MIB" on page 2-2) or by using the SNMP Servlet through a browser connection to your station (see "Create a MIB using the SNMP Servlet" on page 3-21).

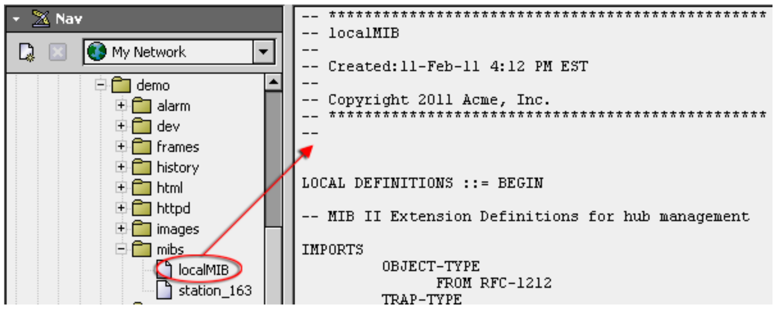
Figure 3-16 *Create a custom MIB using the popup menu on your local device*



Filename `d:\niagara\niagara-3.6.28\stations\demo\mibs\localMIB`

The generated MIB defines the information that can be exposed for a station running the SnmpNetwork. This custom MIB includes entries for the current MIB objects as well as the export tables. After generating the MIB, save the file under any folder that you have read-write access to, as shown in Figure 3-17. After compiling the MIB, an SNMP manager application can use the information to read and write data to and from the station.

Figure 3-17 *Viewing a MIB file in Workbench text file editor view*



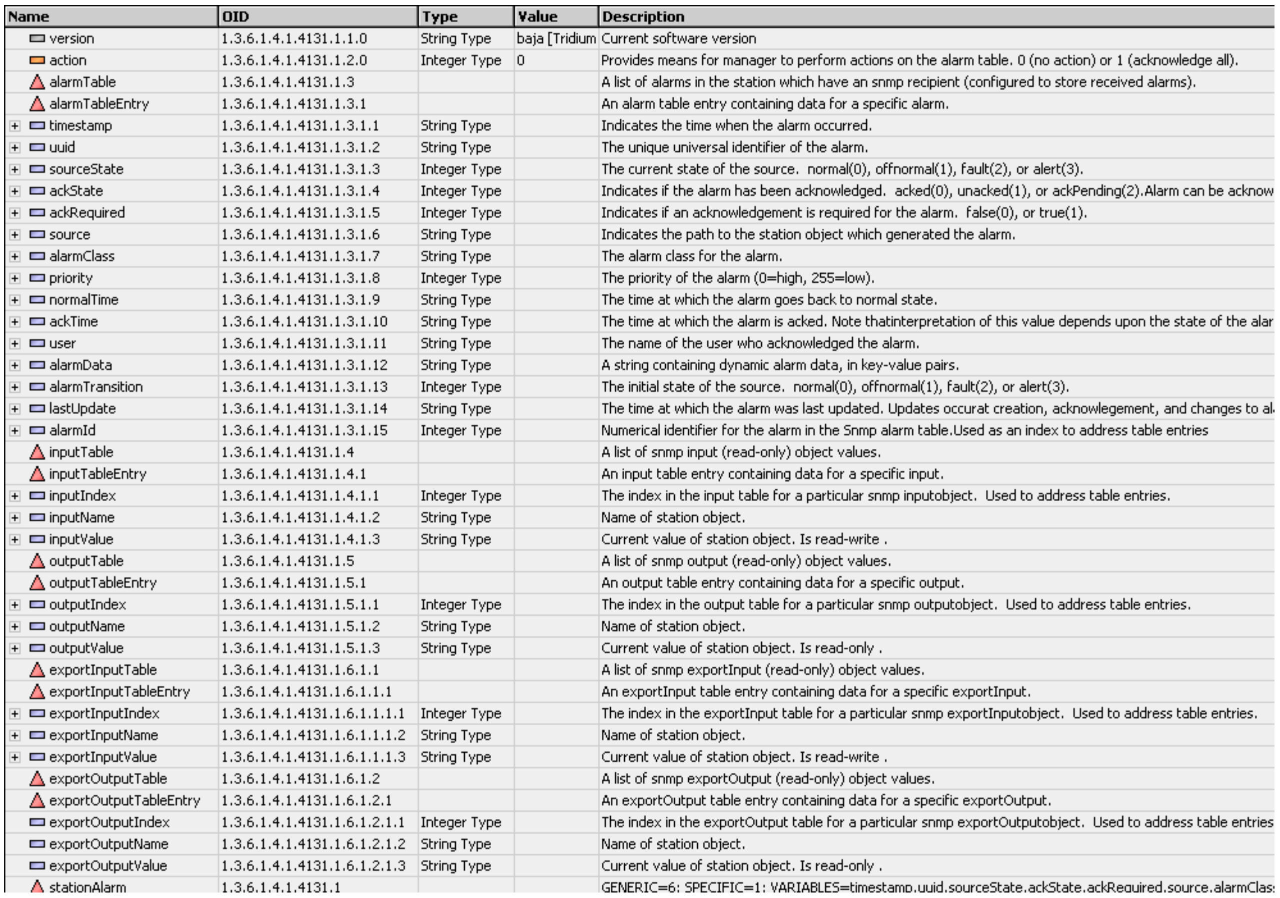
```
-- ****************************************************
-- localMIB
--
-- Created:11-Feb-11 4:12 PM EST
--
-- Copyright 2011 Acme, Inc.
-- ****************************************************
--

LOCAL DEFINITIONS ::= BEGIN

-- MIB II Extension Definitions for hub management

IMPORTS
          OBJECT-TYPE
                 FROM RFC-1212
          TRAP-TYPE
```

"Standard" MIBs are included in the snmp driver module.

Note: *Standard MIBs are those that have been approved by the Internet Architecture Board (IAB). Equipment and software vendors define the private MIBs unilaterally. A branch within the private.enterprises subtree is allocated to each vendor who registers for an enterprise Object Identifier. The distinction between the standard and private MIBs is based on how the variables are defined. The best example of a standard MIB is the RFC1213-MIB (also known as MIB-II).*

Compiling the MIB using the SNMP Point Manager view reveals a list of points, as shown in Figure 3-18. This illustration shows the SNMP Point Manager view displaying five columns. You can show more columns or hide columns, as desired, using the table controls, as described in the *NiagaraAX-3.x Users Guide*. Refer to "About the SNMP point manager view" on page 3-21 for more information about the SNMP Point Manager view.

Figure 3-18 *SNMP MIB compiled*



Name	OID	Type	Value	Description
version	1.3.6.1.4.1.4131.1.1.0	String Type	baja [Tridium	Current software version
action	1.3.6.1.4.1.4131.1.2.0	Integer Type	0	Provides means for manager to perform actions on the alarm table. 0 (no action) or 1 (acknowledge all).
alarmTable	1.3.6.1.4.1.4131.1.3			A list of alarms in the station which have an snmp recipient (configured to store received alarms).
alarmTableEntry	1.3.6.1.4.1.4131.1.3.1			An alarm table entry containing data for a specific alarm.
timestamp	1.3.6.1.4.1.4131.1.3.1.1	String Type		Indicates the time when the alarm occurred.
uuid	1.3.6.1.4.1.4131.1.3.1.2	String Type		The unique universal identifier of the alarm.
sourceState	1.3.6.1.4.1.4131.1.3.1.3	Integer Type		The current state of the source. normal(0), offnormal(1), fault(2), or alert(3).
ackState	1.3.6.1.4.1.4131.1.3.1.4	Integer Type		Indicates if the alarm has been acknowledged. acked(0), unacked(1), or ackPending(2).Alarm can be acknow
ackRequired	1.3.6.1.4.1.4131.1.3.1.5	Integer Type		Indicates if an acknowledgement is required for the alarm. false(0), or true(1).
source	1.3.6.1.4.1.4131.1.3.1.6	String Type		Indicates the path to the station object which generated the alarm.
alarmClass	1.3.6.1.4.1.4131.1.3.1.7	String Type		The alarm class for the alarm.
priority	1.3.6.1.4.1.4131.1.3.1.8	Integer Type		The priority of the alarm (0=high, 255=low).
normalTime	1.3.6.1.4.1.4131.1.3.1.9	String Type		The time at which the alarm goes back to normal state.
ackTime	1.3.6.1.4.1.4131.1.3.1.10	String Type		The time at which the alarm is acked. Note thatinterpretation of this value depends upon the state of the alar
user	1.3.6.1.4.1.4131.1.3.1.11	String Type		The name of the user who acknowledged the alarm.
alarmData	1.3.6.1.4.1.4131.1.3.1.12	String Type		A string containing dynamic alarm data, in key-value pairs.
alarmTransition	1.3.6.1.4.1.4131.1.3.1.13	Integer Type		The initial state of the source. normal(0), offnormal(1), fault(2), or alert(3).
lastUpdate	1.3.6.1.4.1.4131.1.3.1.14	String Type		The time at which the alarm was last updated. Updates occurat creation, acknowlegement, and changes to al
alarmId	1.3.6.1.4.1.4131.1.3.1.15	Integer Type		Numerical identifier for the alarm in the Snmp alarm table.Used as an index to address table entries
inputTable	1.3.6.1.4.1.4131.1.4			A list of snmp input (read-only) object values.
inputTableEntry	1.3.6.1.4.1.4131.1.4.1			An input table entry containing data for a specific input.
inputIndex	1.3.6.1.4.1.4131.1.4.1.1	Integer Type		The index in the input table for a particular snmp inputobject. Used to address table entries.
inputName	1.3.6.1.4.1.4131.1.4.1.2	String Type		Name of station object.
inputValue	1.3.6.1.4.1.4131.1.4.1.3	String Type		Current value of station object. Is read-write .
outputTable	1.3.6.1.4.1.4131.1.5			A list of snmp output (read-only) object values.
outputTableEntry	1.3.6.1.4.1.4131.1.5.1			An output table entry containing data for a specific output.
outputIndex	1.3.6.1.4.1.4131.1.5.1.1	Integer Type		The index in the output table for a particular snmp outputobject. Used to address table entries.
outputName	1.3.6.1.4.1.4131.1.5.1.2	String Type		Name of station object.
outputValue	1.3.6.1.4.1.4131.1.5.1.3	String Type		Current value of station object. Is read-only .
exportInputTable	1.3.6.1.4.1.4131.1.6.1.1			A list of snmp exportInput (read-only) object values.
exportInputTableEntry	1.3.6.1.4.1.4131.1.6.1.1.1			An exportInput table entry containing data for a specific exportInput.
exportInputIndex	1.3.6.1.4.1.4131.1.6.1.1.1.1	Integer Type		The index in the exportInput table for a particular snmp exportInputobject. Used to address table entries.
exportInputName	1.3.6.1.4.1.4131.1.6.1.1.1.2	String Type		Name of station object.
exportInputValue	1.3.6.1.4.1.4131.1.6.1.1.1.3	String Type		Current value of station object. Is read-write .
exportOutputTable	1.3.6.1.4.1.4131.1.6.1.2			A list of snmp exportOutput (read-only) object values.
exportOutputTableEntry	1.3.6.1.4.1.4131.1.6.1.2.1			An exportOutput table entry containing data for a specific exportOutput.
exportOutputIndex	1.3.6.1.4.1.4131.1.6.1.2.1.1	Integer Type		The index in the exportOutput table for a particular snmp exportOutputobject. Used to address table entries
exportOutputName	1.3.6.1.4.1.4131.1.6.1.2.1.2	String Type		Name of station object.
exportOutputValue	1.3.6.1.4.1.4131.1.6.1.2.1.3	String Type		Current value of station object. Is read-only .
stationAlarm	1.3.6.1.4.1.4131.1			GENERIC=6: SPECIFIC=1: VARIABLES=timestamp,uuid,sourceState,ackState,ackRequired,source,alarmClas

The following list describes how the individual MIB entries are used:

- **version (OID: 1.3.6.1.4.1.4131.1.1.0)**
  this read-only string object exposes the current software version that the station is running.

- **action (OID: 1.3.6.1.4.1.4131.1.2.0)**
  this read-write integer object allows the manager to perform certain actions on the Alarm Table. A value of 0 causes no action to be performed on the Alarm Table (remember to always set the value back to 0 after performing any action, otherwise the action will be performed repeatedly on any new alarm entries). A value of 1 causes all of the alarm entries in the Alarm Table to be acknowledged (and also in the Niagara Alarming Service if the SnmpNetwork for the station has the 'Synchronize Alarm Acks' property enabled - see SnmpNetwork Configuration). A value of 2 causes all of the alarms to be cleared (the corresponding alarms in the Niagara Alarming Service will be acknowledged if unacknowledged and if the SnmpNetwork for the station has the 'Synchronize Alarm Acks' property enabled). A value of 3 causes all acknowledged alarms in the Alarm Table to be cleared. Any other value (outside the range of 0-3) is not valid for this object, and will be treated as no action.
- **alarmTable (OID: 1.3.6.1.4.1.4131.1.3)**
  this sequence of alarmTableEntries represents the Alarm Table that is dynamically created by the station whenever an alarm is generated in the station and routed to a Alarm Class that has an SnmpRecipient object linked to it. The Alarm Table has a buffer size of 50 alarm entries. Alarm entries that are not cleared from the Alarm Table when the buffer becomes full will be removed on a FIFO (First In, First Out) basis as new alarm entries are added.
- **alarmTableEntry (OID: 1.3.6.1.4.1.4131.1.3.1)**
  this represents an entry within the alarmTable containing the following data (all fields of an Alarm Record):
  - alarmId (OID: 1.3.6.1.4.1.4131.1.3.1.1)
    this read-only integer object exposes the numerical identifier automatically assigned to the alarm entry in the Alarm Table. It is used as an index to address table entries.
  - ackState (OID: 1.3.6.1.4.1.4131.1.3.1.2)
    this read-write integer object allows the manager to acknowledge and view the acknowledge state of the alarm entry. A value of 1 means that the alarm is unacknowledged. Setting the value to 2 acknowledges the alarm. After setting the value to 2, the manager should re-read the value of this object again to verify that the acknowledgement occurred (the value will be 2 when acknowledged).
  - ackRequired (OID: 1.3.6.1.4.1.4131.1.3.1.3)
    this read-only integer object exposes whether an acknowledgement is required for this alarm entry. A value of 1 (true) indicates that an acknowledgement is required, while a value of 2 (false) indicates that an acknowledgement is not required.
  - timestamp (OID: 1.3.6.1.4.1.4131.1.3.1.4)
    this read-only string object exposes the time when the alarm occurred (in the Niagara station's timestamp form).
  - recordType (OID: 1.3.6.1.4.1.4131.1.3.1.5)
    this read-only string object exposes the type of the Niagara alarm record.
  - source (OID: 1.3.6.1.4.1.4131.1.3.1.6)
    this read-only string object exposes the path to the station object which generated the alarm.
  - alarmFlags (OID: 1.3.6.1.4.1.4131.1.3.1.7)
    this read-only string object exposes the flags associated with the alarm.
  - alarmData (OID: 1.3.6.1.4.1.4131.1.3.1.8)
    this read-only string object exposes the alarm data.
  - uuid (OID: 1.3.6.1.4.1.4131.1.3.1.9)
    this read-only string object exposes the unique universal identifier of the alarm.
  - alarmClass (OID: 1.3.6.1.4.1.4131.1.3.1.10)
    this read-only string object exposes the path to the Niagara Alarm Class of the alarm.
  - priority (OID: 1.3.6.1.4.1.4131.1.3.1.11)
    this read-only integer object exposes the priority of the alarm.
  - ackTime (OID: 1.3.6.1.4.1.4131.1.3.1.12)
    this read-only string object exposes the time at which the alarm was acked (in the Niagara station's timestamp form). Note that interpretation of this value depends upon the state of the alarm.
  - user (OID: 1.3.6.1.4.1.4131.1.3.1.13)
    this read-only string object exposes the name of the user who acknowledged the alarm.
- **inputTable (OID: 1.3.6.1.4.1.4131.1.4)**
  this sequence of inputTableEntries represents the Input Table that is dynamically created by the station whenever an input-type SnmpAgent proxy point is created in the station under an SnmpAgent.
- **inputTableEntry (OID: 1.3.6.1.4.1.4131.1.4.1)**
  represents an entry within the inputTable containing the following data:

- inputIndex (OID: 1.3.6.1.4.1.4131.1.4.1.1)
  this read-only integer object exposes the index into the Input Table for a particular input-type SnmpAgent proxy point's value. It is used as an index to address Input Table entries.
- inputName (OID: 1.3.6.1.4.1.4131.1.4.1.2)
  this read-only string object exposes the name of the input-type SnmpAgent proxy point supplying the value of this Input Table entry.
- inputValue (OID: 1.3.6.1.4.1.4131.1.4.1.3)
  this read-write string object allows the value stored in the corresponding input-type SnmpAgent proxy point to be read or written. All values are read and written as string types, and the input-type SnmpAgent proxy point on the station itself will handle conversions if necessary (i.e. converting from string to integer or boolean type)

- **outputTable (OID: 1.3.6.1.4.1.4131.1.5)**
  this sequence of outputTableEntries represents the Output Table that is dynamically created by the station whenever an output-type SnmpAgent proxy point is created in the station.
- **outputTableEntry (OID: 1.3.6.1.4.1.4131.1.5.1)**
  represents an entry within the outputTable containing the following data:
  - outputIndex (OID: 1.3.6.1.4.1.4131.1.5.1.1)
    this read-only integer object exposes the index into the Output Table for a particular output-type SnmpAgent proxy point's value. It is used as an index to address Output Table entries.
  - outputName (OID: 1.3.6.1.4.1.4131.1.5.1.2)
    this read-only string object exposes the name of the output-type SnmpAgent proxy point supplying the value of this Output Table entry.
  - outputValue (OID: 1.3.6.1.4.1.4131.1.5.1.3)
    this read-only string object exposes the value stored in the corresponding output-type SnmpAgent proxy point. All values are read as string types, and thus the manager must handle conversions if necessary (i.e. converting from string to integer or boolean type).

- **exportInputTable (OID: 1.3.6.1.4.1.4131.1.6.1.1)**
  this sequence of exportInputTableEntries represents the exportInput Table that is dynamically created by the station whenever an exportInput-type SnmpAgent proxy point is created in the station under an SnmpAgent.
- **exportInputTableEntry (OID: 1.3.6.1.4.1.4131.1.6.1.1.1)**
  represents an entry within the exportInputTable containing the following data:
  - **exportI**nputIndex (OID: 1.3.6.1.4.1.4131.6.1.1.1.1)
    this read-only integer object exposes the index into the exportInput Table for a particular exportInput-type SnmpAgent proxy point's value. It is used as an index to address exportInput Table entries.
  - **exportI**nputName (OID: 1.3.6.1.4.1.4131.1.6.1.1.1.2)
    this read-only string object exposes the name of the exportInput-type SnmpAgent proxy point supplying the value of this exportInput Table entry.
  - **exportI**nputValue (OID: 1.3.6.1.4.1.4131.1.6.1.1.1.3)
    this read-write string object allows the value stored in the corresponding exportInput-type SnmpAgent proxy point to be read or written. All values are read and written as string types, and the exportInput-type SnmpAgent proxy point on the station itself will handle conversions if necessary (i.e. converting from string to integer or boolean type)

- **exportOutputTable (OID: 1.3.6.1.4.1.4131.1.6.1.2)**
  this sequence of exportOutputTableEntries represents the exportOutput Table that is dynamically created by the station whenever an exportOutput-type SnmpAgent proxy point is created in the station.
- **exportOutputTableEntry (OID: 1.3.6.1.4.1.4131.1.6.1.2.1)**
  represents an entry within the exportOutputTable containing the following data:
  - **exportO**utputIndex (OID: 1.3.6.1.4.1.4131.1.6.1.1.1.1)
    this read-only integer object exposes the index into the exportOutput Table for a particular exportOutput-type SnmpAgent proxy point's value. It is used as an index to address exportOutput Table entries.
  - **exportO**utputName (OID: 1.3.6.1.4.1.4131.1.6.1.1.1.2)
    this read-only string object exposes the name of the exportOutput-type SnmpAgent proxy point supplying the value of this exportOutput Table entry.
  - **exportO**utputValue (OID: 1.3.6.1.4.1.4131.1.6.1.1.1.3)

this read-only string object exposes the value stored in the corresponding exportOutput-type SnmpAgent proxy point. All values are read as string types, and thus the manager must handle conversions if necessary (i.e. converting from string to integer or boolean type).

- **stationAlarm (OID: 1.3.6.1.4.1.4131.1)**
  this SNMP trap-type defines the form of outgoing SNMPv1 trap messages sent by the station. For more details, see Generating SNMP Traps.

About the SNMP Servlet

You can use a browser connection to create a custom MIB. You need to connect to a station that contains your desired local device.

Create a MIB using the SNMP Servlet

You can create an SNMP MIB by connecting to a station that has SNMP licensed and configured.

Step 1 Type the address of your station followed by "`/snmp`" in the browser address bar, for example:

```
http://138.18.60.188/snmp
```

- If you are not already logged to the station, the **`Login`** dialog box displays.
  Log in to the station, as required.
- When you are logged into the station, the MIB file appears in the browser.

Step 2 From the browser main menu, select **`File > Save`** (or similar menu selections, as available in your browser) to save your MIB file to the desired location.

The MIB file is available for use if you can connect to it from your station.

Types of manager views

Manager views provide ways for you to discover, add, match, or simply view snmp points or objects in various ways. The following sections provide descriptions of the different manager views that are available on the Agent (local device) and Manager (client) devices.

- "About the SNMP point manager view" on page 3-21
- "About the Snmp Table Manager view" on page 3-24
- "About the Snmp Trap Manager view" on page 3-25
- "About the Snmp Agent Point Manager view" on page 3-25
- "About the Snmp Export Manager (SnmpExportTable)" on page 3-26
- "About the Snmp Object Manager view" on page 3-27

About the SNMP point manager view

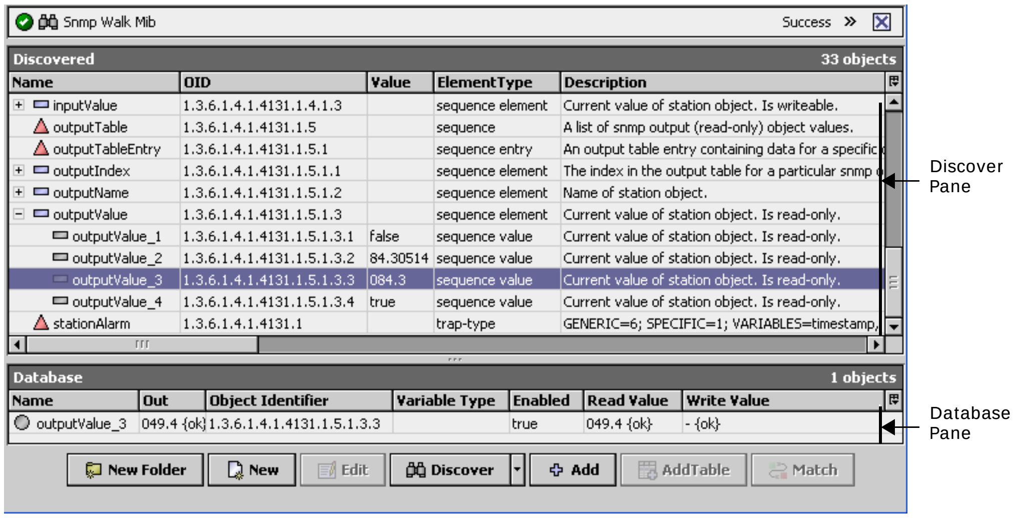
The SNMP point manager view is the default view on Snmp Point Device Extension. It is used to compile and display the contents of MIB files for easy creation of SNMP proxy points that represent data values in the actual SNMP device (based on the information contained in the compiled MIB file). It is also used for viewing, storing, and removing SNMP trap and notification types learned for an SnmpDevice from the compiled MIB file.

The SNMP point manager view is basically the same as the standard point manager view that is used for most Niagara drivers and is described in detail in the *NiagaraAX-3.x Drivers Guide*.

The SNMP point manager view splits into two panes when you do a "Discover" (see Figure 3-19).

- **Discover pane**
  This pane displays MIB entries from the last compiled MIB file.
- **Database pane**
  This pane displays the points that are added to your station database.

Figure 3-19 SNMP point manager view



Name	OID	Value	ElementType	Description
inputValue	1.3.6.1.4.1.4131.1.4.1.3		sequence element	Current value of station object. Is writeable.
outputTable	1.3.6.1.4.1.4131.1.5		sequence	A list of snmp output (read-only) object values.
outputTableEntry	1.3.6.1.4.1.4131.1.5.1		sequence entry	An output table entry containing data for a specific
outputIndex	1.3.6.1.4.1.4131.1.5.1.1		sequence element	The index in the output table for a particular snmp
outputName	1.3.6.1.4.1.4131.1.5.1.2		sequence element	Name of station object.
outputValue	1.3.6.1.4.1.4131.1.5.1.3		sequence element	Current value of station object. Is read-only.
outputValue_1	1.3.6.1.4.1.4131.1.5.1.3.1	false	sequence value	Current value of station object. Is read-only.
outputValue_2	1.3.6.1.4.1.4131.1.5.1.3.2	84.30514	sequence value	Current value of station object. Is read-only.
outputValue_3	1.3.6.1.4.1.4131.1.5.1.3.3	084.3	sequence value	Current value of station object. Is read-only.
outputValue_4	1.3.6.1.4.1.4131.1.5.1.3.4	true	sequence value	Current value of station object. Is read-only.
stationAlarm	1.3.6.1.4.1.4131.1		trap-type	GENERIC=6; SPECIFIC=1; VARIABLES=timestamp,

Name	Out	Object Identifier	Variable Type	Enabled	Read Value	Write Value
outputValue_3	049.4 {ok}	1.3.6.1.4.1.4131.1.5.1.3.3		true	049.4 {ok}	- {ok}

In addition to the standard point manager functions, the SNMP point manager view has some SNMP-specific functions that are described in the following sections:

- "About SNMP point discovery" on page 3-23
- "Loading the MIB" on page 3-23
- "Configuring dependent MIBs" on page 3-24

About the SNMP point manager Discover pane

The discover pane displays MIB entries from the last compiled MIB file. These entries provide a template for creating and pre-configuring SNMP proxy points for the SnmpDevice (in particular, it automatically sets up such SNMP proxy point properties as the 'Object Identifier', or OID). The columns of the table include:

- **Name**
  The name of the OBJECT-TYPE of the MIB entry. This name will be used to give a name to any created SNMP proxy points based on the entry.
- **OID**
  The Object Identifier of the given entry. This OID will be pre-configured in the 'Object Identifier' property of any created SNMP proxy points based on the entry.
- **Type**
  The type of object given in the SYNTAX section of the MIB entry. Common examples are INTEGER, DisplayString, Gauge, IpAddress, Counter, TimeTicks, etc. This type is used to pre-configure the 'Variable Type' property of any created writable SNMP proxy points based on the entry.
- **Value**
  This column displays the point value. When the MIB is loaded but not "walked" the value column is empty.
- **Access**
  The accessibility of the MIB entry (read-only, read-write, not-accessible). Used to determine if and what type (read-only or writable) of SNMP proxy points can be created for the given entry.
- **Status**
  The current relevance of the MIB entry: mandatory, current, optional, deprecated, or obsolete. For display purposes only, does not affect creation of SNMP proxy points based on the entry.
- **ElementType**
  This column is only used internally by Niagara during creation of SNMP proxy points based on selected entries in the table. It displays whether the MIB entry is a sequence element, trap-type, notification-type, or non-sequence element.
- **Description**
  Displays a summary of the function of the given MIB entry.

About the SNMP point manager Database pane

The SNMP point manager Database pane is located below the Discover pane. This pane lists the proxy points that are currently in the database. You may edit some of the point values, depending on the value type, by double clicking on the desired row and using the `Edit` dialog box. The columns of the table include:

- **Path**
  This column displays the ORD path, starting from the station database to the listed point.
- **Name**
  This is the displayed point name.
- **Type**
  This is the type of object given in the SYNTAX section of the MIB entry. Common examples are Integer, DisplayString, Gauge, IpAddress, Counter, TimeTicks, etc. This type is used to pre-configure the 'Variable Type' property of any created writable SNMP proxy points based on the entry.
- **Out**
  This is the value of the point
- **Object Identifier**
  The Object Identifier of the given point. This OID will be pre-configured in the 'Object Identifier' property of any created SNMP proxy points based on the entry.
- **Variable Type**
  This is the variable type of the proxy point. You can edit the variable type for input values but cannot edit the variable type for output values.
- **Enabled**
  This property allows you to set the proxy point in service (with a `true` value) or to set it out of service (with a `false` value).
- **Device Facets**
  This property represents the device proxy point facets for how the value should be displayed in Niagara.
- **Facets**
  This property represents the parent Niagara proxy point's facets, for how the value should be displayed in Niagara.
- **Conversion**
  This property specifies the conversion to use between the "read value" (in Device Facets) and the parent point facets, where "Default" is typically used.
- **Read Value**
  (read only) This is the last value read from the device, expressed in device facets.
- **Write Value**
  (read only) This is applies only if the point is writable. This is the last value written, using device facets.
- **Tuning Policy Name**
  This property specifies the SNMP service type to use when binding to this item.
- **Value**
  This column displays the point value. When the MIB is loaded but not "walked" the value column is empty.

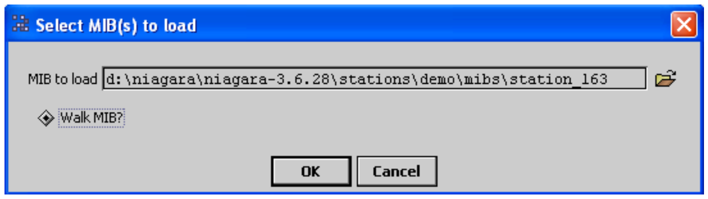
About SNMP point discovery

The SNMP point discovery (or "learn") process includes two steps:

- **Loading the MIB**
  "Loading the MIB" is the process where the point manager compiles, reads, and displays points in the table pane of the SNMP point manager view. See "Loading the MIB" on page 3-23 for details.
- **Walking the MIB**
  "Walking the MIB is where communication takes place between an agent (client) application and a manager application to discover the point values for those loaded point types that have values available.

Loading the MIB When you click the **`Discover`** button, the SNMP point manager opens the **`Select MIBs to load`** *dialog box.*

Figure 3-20 **`Select MIBs to load`** *dialog box*



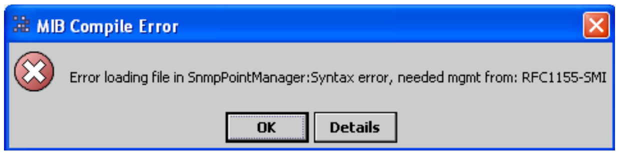
MIB to load `d:\niagara\niagara-3.6.28\stations\demo\mibs\station_163`

The “MIB to load” field allows for specifying the MIB and populating the point table based on the MIB file that you select. When a MIB file is “loaded”, the MIB file and any dependent files are read into memory. The object types that are available for a device are limited to those types that are defined in the MIB file. Loading the MIB is a client-side operation that does not require remote communication. If you do not select the “Walk the MIB?” option, no data values are determined for point types that are defined in the MIB. This might be an option that you use when you are developing an application offline. After developing your application, you can “walk” the MIB when you can establish communication between the SNMP agent and manager applications.

Note: *The resulting table contains only MIB entries for the most recent MIB file to be compiled and loaded.*

If there are errors when trying to compile the MIB files that you selected, an error message displays to indicate the errors encountered. For example, a missing dependent MIB file would cause an exception pop-up similar to the one shown in Figure 3-21

Figure 3-21 *MIB compile error message*

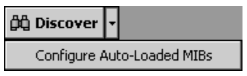

MIB Compile Error

Error loading file in SnmpPointManager:Syntax error, needed mgmt from: RFC1155-SMI

If no errors are encountered after issuing the LoadMIB command, the MIB entries contained in the loaded MIB file appear in the table (as shown in Figure 3-18 and Figure 3-19).

Configuring dependent MIBs Dependent MIBs may be required by the primary MIB. Any MIBs that are not already available in the snmp module may be placed in one or more accessible folders.

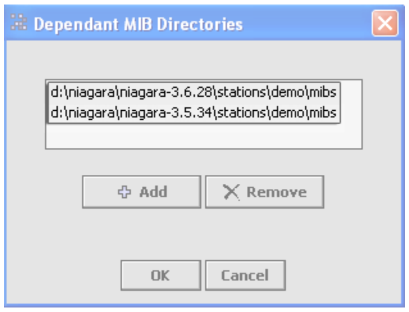
The **`Dependent MIB Directories`** dialog box provides a way to specify MIB file locations that satisfy dependencies of the primary MIB file.

This dialog box displays when you select **`Configure MIB Path`** 

from the right corner of the **`Discover`** button.

This is a time-saver if you have one or more locations where you store dependency MIBs. Once the folder locations are specified in this dialog box (as shown in Figure 3-22) they are loaded any time that a **`Load MIB`** type command is issued (for example, **`Load MIB and Walk MIB`**).

Figure 3-22 *Dependent MIB Directories*


Dependant MIB Directories

d:\niagara\niagara-3.6.28\stations\demo\mibs
d:\niagara\niagara-3.5.34\stations\demo\mibs

Note: *The Dependent MIBs list configuration is saved as a user option. Therefore, running Workbench from a different platform may present you with a different list of MIB directories.*

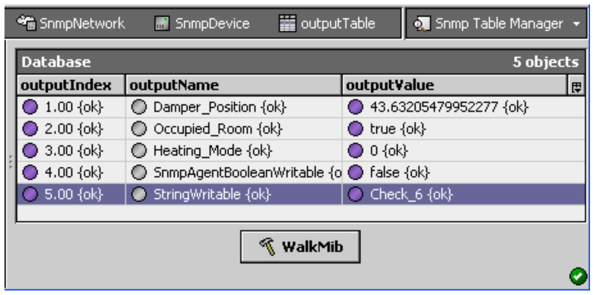
The **`Dependent MIB Directories`** dialog box allows you to add and remove MIB directories from a list, as described below:

- **Add** MIB directories to the list
  Click the **Add** button to navigate to a directory to add to the list.
- **`Remove`** MIB directories from the directories list
  Use the **`Remove`** button to remove any selected directories from the list.

About the Snmp Table Manager view

The SNMP Table Manager view is the default view on the Snmp Table component. The purpose of this view is to provide a summary view of data. Three columns: Index, Name, and Value display read-only values. Table objects are visible in the Point Manager view. Double-click on the table object to view the data in the Snmp Table Manager view.

Figure 3-23 *Snmp Table Manager view*



Database — 5 objects

outputIndex	outputName	outputValue
1.00 {ok}	Damper_Position {ok}	43.63205479952277 {ok}
2.00 {ok}	Occupied_Room {ok}	true {ok}
3.00 {ok}	Heating_Mode {ok}	0 {ok}
4.00 {ok}	SnmpAgentBooleanWritable {o	false {ok}
5.00 {ok}	StringWritable {ok}	Check_6 {ok}

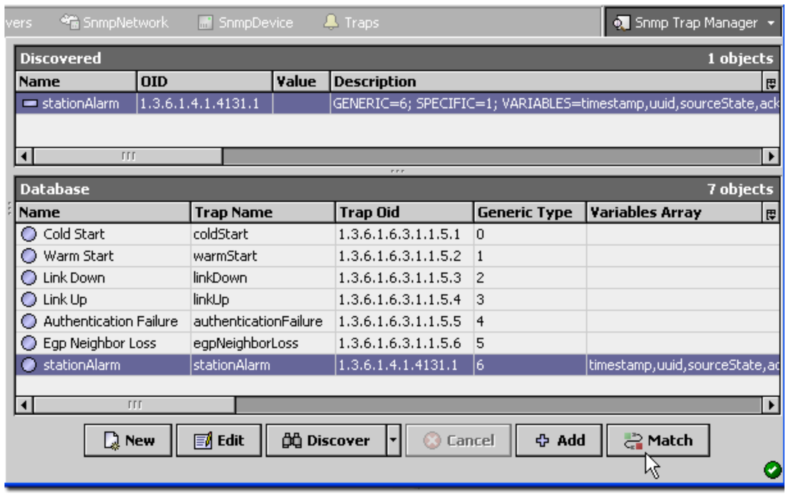
Use the **`WalkMib`** button at the bottom of the view to walk the MIB.

About the Snmp Trap Manager view

The SNMP Trap Manager view is the default view on the Snmp Alarm Device Extension. This view provides a summary view of data and allow you to discover, compile, display, and store traps from MIB files for an SnmpDevice.

You can use the controls along the bottom of the view to work with traps. Use the **`Add`** and **`Match`** buttons to move "discovered" traps from the top (Discovered) pane to the lower (Database) pane. Use the **`New`** button to create new trap types and the **`New`** button to change property values in an existing trap in the database pane. Click the **`Discover`** button to populate the Discovered pane. As with the Snmp Point Manager view, the **`Discover`** button has an additional drop-down button that you can click to open the **`Dependent MIB Directories`** dialog box. See "About SNMP point discovery" on page 3-23 for more information about discovery and setting the MIB directory.

Figure 3-24 *Snmp Trap Manager view*



Discovered — 1 objects

Name	OID	Value	Description
stationAlarm	1.3.6.1.4.1.4131.1		GENERIC=6; SPECIFIC=1; VARIABLES=timestamp,uuid,sourceState,ack

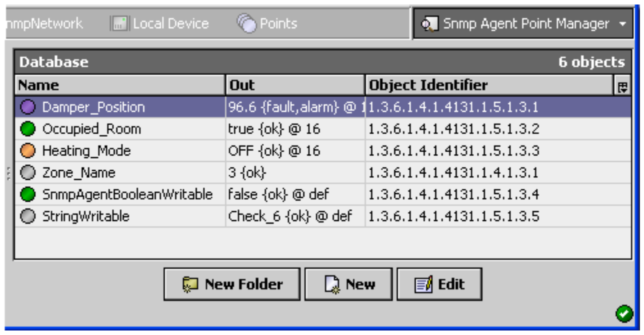
Database — 7 objects

Name	Trap Name	Trap Oid	Generic Type	Variables Array
Cold Start	coldStart	1.3.6.1.6.3.1.1.5.1	0	
Warm Start	warmStart	1.3.6.1.6.3.1.1.5.2	1	
Link Down	linkDown	1.3.6.1.6.3.1.1.5.3	2	
Link Up	linkUp	1.3.6.1.6.3.1.1.5.4	3	
Authentication Failure	authenticationFailure	1.3.6.1.6.3.1.1.5.5	4	
Egp Neighbor Loss	egpNeighborLoss	1.3.6.1.6.3.1.1.5.6	5	
stationAlarm	stationAlarm	1.3.6.1.4.1.4131.1	6	timestamp,uuid,sourceState,ac

About the Snmp Agent Point Manager view

The SNMP Agent Point Manager view is the default view on the Snmp Agent Point Device Extension, which is located under the Snmp Local (Agent) Device. This view provides a summary view of data and allows you to add new points to the view that can be exposed on the SnmpNetwork.

Figure 3-25 *Snmp Agent Point Manager view*



Database 6 objects

Name	Out	Object Identifier
Damper_Position	96.6 {fault,alarm} @ 1	1.3.6.1.4.1.4131.1.5.1.3.1
Occupied_Room	true {ok} @ 16	1.3.6.1.4.1.4131.1.5.1.3.2
Heating_Mode	OFF {ok} @ 16	1.3.6.1.4.1.4131.1.5.1.3.3
Zone_Name	3 {ok}	1.3.6.1.4.1.4131.1.4.1.3.1
SnmpAgentBooleanWritable	false {ok} @ def	1.3.6.1.4.1.4131.1.5.1.3.4
StringWritable	Check_6 {ok} @ def	1.3.6.1.4.1.4131.1.5.1.3.5

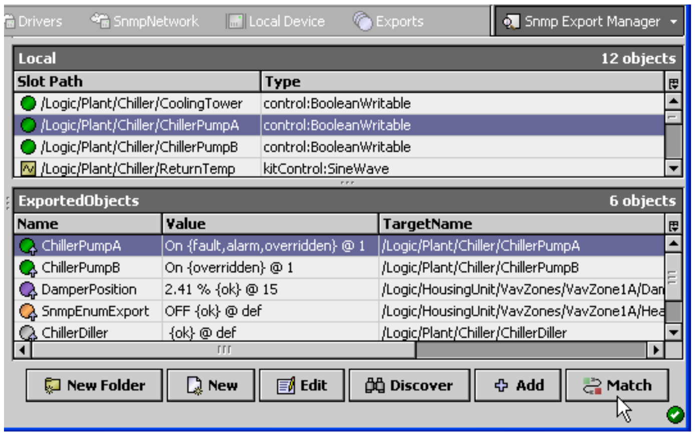
The Snmp Agent Point Manager works differently than Snmp Point Manager. For example, the Snmp Agent Point Manager does not have a **`Discover`** button for adding proxy points. Typically, you add new points by copying Agent Points from the SNMP palette or by using the **`New`** button at the bottom of the view. Use the **`New Folders`** button to organize and add hierarchy to your point collection. Use the **`Edit`** button to change a selected point's property values.

About the Snmp Export Manager (SnmpExportTable)

The SNMP Export Manager view is the default view on Snmp Export Table Extension.

The SNMP Export Manager view is similar to the Snmp Point Manager view, however, in this view, the **`Discover`** button launches the **`Bql Query Builder`** dialog box, instead initiating a MIB loading and walking process. The **`Bql Query Builder`** dialog box lets you "discover" components to add to the Local (top) pane, in a manner similar to other discoveries. However, in this case you are discovering points that are on your local device. Once the points are discovered, they can be selected and moved to the ExportedObjects (lower) pane using the **`Add`** or **`Match`** button. See the *NiagaraAX User Guide* for more information about the using the **`Bql Query Builder`**.

Figure 3-26 *SNMP Export Manager view*



Local 12 objects

Slot Path	Type
/Logic/Plant/Chiller/CoolingTower	control:BooleanWritable
/Logic/Plant/Chiller/ChillerPumpA	control:BooleanWritable
/Logic/Plant/Chiller/ChillerPumpB	control:BooleanWritable
/Logic/Plant/Chiller/ReturnTemp	kitControl:SineWave

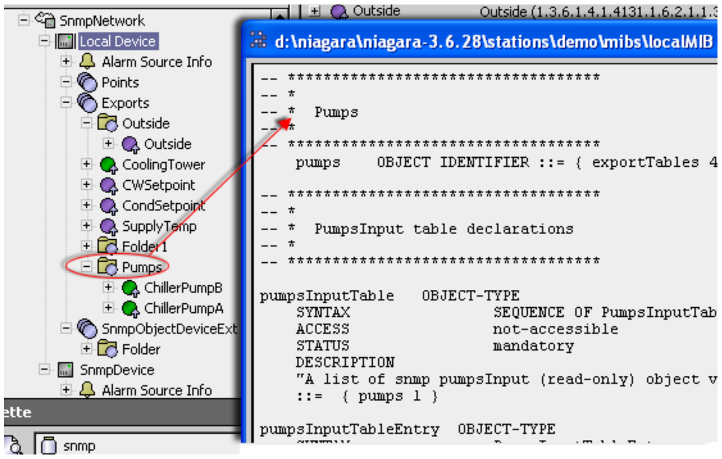
ExportedObjects 6 objects

Name	Value	TargetName
ChillerPumpA	On {fault,alarm,overridden} @ 1	/Logic/Plant/Chiller/ChillerPumpA
ChillerPumpB	On {overridden} @ 1	/Logic/Plant/Chiller/ChillerPumpB
DamperPosition	2.41 % {ok} @ 15	/Logic/HousingUnit/VavZones/VavZone1A/Dan
SnmpEnumExport	OFF {ok} @ def	/Logic/HousingUnit/VavZones/VavZone1A/Hea
ChillerDiller	{ok} @ def	/Logic/Plant/Chiller/ChillerDiller

After adding points to the ExportedObjects pane, the points appear under the Exports (Snmp Export Table) node in the nav tree.

You can use the **`New Folder`** button along the bottom of the view, to add an Export Folder component. The default view of these folders is the Snmp Export Manager. You can nest these folders and move points between folders; OIDs are adjusted automatically. Each folder is represented in a locally-generated MIB as an input and output table, the same way that points in the Points folder are represented. See the following illustration that shows an Snmp Export Folder in the nav tree and as represented in a MIB.

Figure 3-27 *SNMP Export Folder shows up in the MIB file*



```
-- ************************************
-- *
-- *  Pumps
-- *
-- ************************************
    pumps     OBJECT IDENTIFIER ::= { exportTables 4

-- ************************************
-- *
-- *  PumpsInput table declarations
-- *
-- ************************************

pumpsInputTable   OBJECT-TYPE
    SYNTAX                    SEQUENCE OF PumpsInputTab
    ACCESS                    not-accessible
    STATUS                    mandatory
    DESCRIPTION
    "A list of snmp pumpsInput (read-only) object v
    ::=  { pumps 1 }

pumpsInputTableEntry  OBJECT-TYPE
```

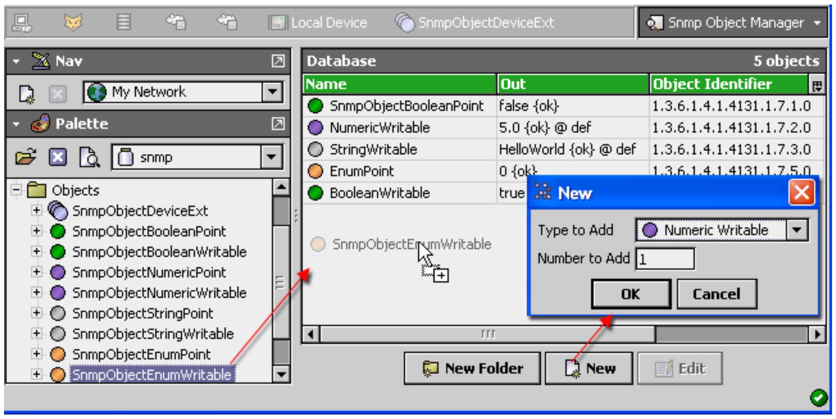
About the Snmp Object Manager view

The SNMP Object Manager view is the default view on the Snmp Object Device Extension. The view shows object points in a table view. You can use the buttons along the bottom of the view to add folders or Snmp Object points directly under the Snmp Object Device Extension.

Note: *Snmp Objects allow you to expose any snmp ASN data type instead of just the string type supported in the tables.*

You can also drag points from the Objects folder (in the snmp palette) onto the view to add them, as shown in Figure 3-28.

Figure 3-28 *SNMP Object Manager view*



Database 5 objects

Name	Out	Object Identifier
SnmpObjectBooleanPoint	false {ok}	1.3.6.1.4.1.4131.1.7.1.0
NumericWritable	5.0 {ok} @ def	1.3.6.1.4.1.4131.1.7.2.0
StringWritable	HelloWorld {ok} @ def	1.3.6.1.4.1.4131.1.7.3.0
EnumPoint	0 {ok}	1.3.6.1.4.1.4131.1.7.5.0
BooleanWritable	true	

New

Type to Add	Numeric Writable
Number to Add	1

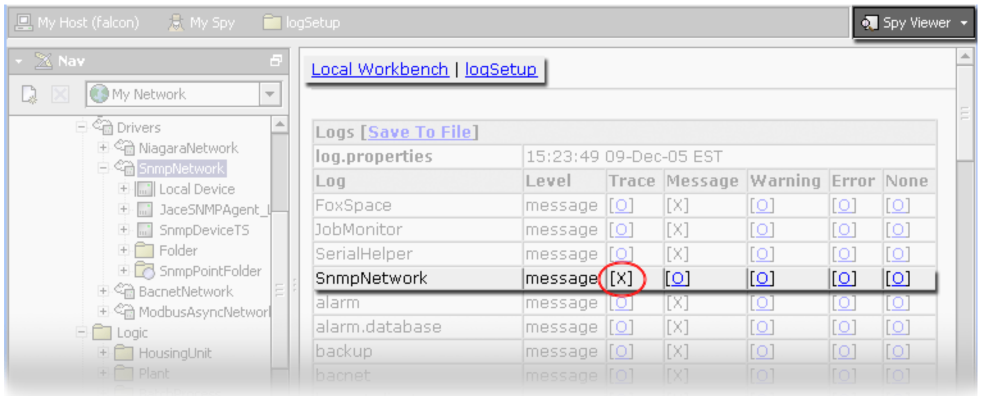
After adding points to the Snmp Object database pane, the points appear under the Snmp Object Device Extension node in the nav tree.

Troubleshooting and debugging

You can view debugging messages specific to the SNMP driver by configuring the diagnostics - logSetup view of a running station with an SnmpNetwork (see example below).

A single **`log`** is created for an SnmpNetwork object existing in a running station (named `SnmpNetwork`). This log is used for displaying SNMP-specific information to standard output. If you enable **`Trace`** for this log, it displays all levels of debug information specific to the SNMP driver to the standard output, including SNMP communication (such as byte requests/responses).

Note: *Enabling Trace for this log may lower performance due to the extra overhead of writing to standard output. You should use this function when you need to debug your application and then disable it when you are finished. The* **`SnmpNetwork`** *log only shows debug information particular to its own network (and devices/proxy points underneath). You can use the Trace feature to verify communications and polling of your devices/proxy points.*



Logs [Save To File]

log.properties	15:23:49 09-Dec-05 EST					
Log	Level	Trace	Message	Warning	Error	None
FoxSpace	message	[O]	[X]	[O]	[O]	[O]
JobMonitor	message	[O]	[X]	[O]	[O]	[O]
SerialHelper	message	[O]	[X]	[O]	[O]	[O]
SnmpNetwork	message	[X]	[O]	[O]	[O]	[O]
alarm	message	[O]	[X]	[O]	[O]	[O]
alarm.database	message	[O]	[X]	[O]	[O]	[O]
backup	message	[O]	[X]	[O]	[O]	[O]
bacnet	message	[O]	[X]	[O]	[O]	[O]

Figure 3-29 *SNMP log setup*

CHAPTER 4

Plugin Guides

Plugins provide a visualization of a Component. There are many ways to view plugins. One way is directly in the tree. In addition, you can right-click on an item and select one of its views. Plugins provide views of Components. You can access documentation on a Plugin by selecting **Help > On View** (F1) from the menu or pressing F1 while the Plugin is selected.

Types of Snmp plugins

Following, is a list of Snmp plugins:

- SnmpAgentPointManager
- SnmpDeviceManager
- SnmpExportManager
- SnmpObjectManager
- SnmpPointManager
- SnmpTableManager
- SnmpTrapManager

snmp-SnmpAgentPointManager

Use the SnmpAgentPointManager to create, edit, access, and delete SnmpAgent proxy points under a SnmpAgent. The SnmpAgentPointManager is the default view for the SnmpAgentPointDeviceExt (Points container) under an SnmpAgent. The SnmpAgentPointManager is also the default view for any SnmpAgentPointFolder under the Points container of an SnmpAgent.

To view, right-click a SnmpAgentPointDeviceExt or SnmpAgentPointFolder and select **Views > SNMP Agent Point Manager**

snmp-SnmpDeviceManager

Use the SnmpDeviceManager to create, edit, and view both SnmpDevices and SnmpAgents under a SnmpNetwork. The SnmpDeviceManager is the default view on the SnmpNetwork. To view, right-click SnmpNetwork and select **Views > SNMP Device Manager**

snmp-SnmpExportManager

The SnmpExportManager is the default view of the Snmp Export Table component. The view has an upper (discovery) and lower (database) pane that you use for discovering and adding points or tables for exporting snmp values. Use this view to discover and add points and tables to your Snmp Export Table under the Snmp Local Device. Also see, "About the Snmp Export Manager (SnmpExportTable)" on page 3-26.

snmp-SnmpObjectManager

The SnmpObjectManager is the default view of the SnmpObjectDeviceExt (see snmp-SnmpObjectDeviceExt). This view provides a single database pane that allows you to add SnmpObject types and folders using the buttons at the bottom of the view. The added Snmp Objects allow you to expose any snmp ASN data type instead of just the string type supported in the tables. The SnmpObjectDeviceExt is available under the Objects folder in the Snmp palette.

snmp-SnmpPointManager

Use the SnmpPointManager to create, edit, access, and delete SNMP proxy points under a SnmpDevice. The SnmpPointManager is the default view for the SnmpPointDeviceExt (Points container) under an SnmpDevice. The SnmpPointManager is also the default view for any SnmpPointFolder under the Points container of an SnmpDevice.

To view, right-click a SnmpPointDeviceExt or SnmpPointFolder and select **`Views > SNMP Point Manager`**

snmp-SnmpTableManager

The SnmpTableManager is the default view of any Snmp Table (outputTable or exportTable). The view displays an ordered sequence of rows with the output index number, name and value in separate columns. The view has a **`WalkMib`** button for updating values by walking the MIB.

snmp-SnmpTrapManager

Use the SnmpTrapManager to discover, compile, display, and store traps from MIB files for an SnmpDevice. The SnmpTrapManager is the default view for an SnmpDevice's TrapTable slot (default name `Trap Types`).

To view, right-click the TrapTable of an SnmpDevice and select **`Views > SNMP Trap Manager`**

CHAPTER 5

Using the Component Guides

The Component Guides provides help on common components. Summary information is provided on components specific to the snmp module, listed in alphabetical order as follows:

- snmp-MIBListTable
- snmp-SnmpAgent
- snmp-SnmpAgentBooleanProxyExt
- snmp-SnmpAgentNumericProxyExt
- snmp-SnmpAgentPointDeviceExt
- snmp-SnmpAgentPointFolder
- snmp-SnmpAgentStringProxyExt
- snmp-SnmpAlarmDeviceExt
- snmp-SnmpBooleanExport
- snmp-SnmpBooleanProxyExt
- snmp-SnmpEnumExport
- snmp-SnmpExportTable
- snmp-SnmpNumericExport
- snmp-SnmpObjectDeviceExt
- snmp-SnmpSequence
- snmp-SnmpStringExport
- snmp-SnmpTable
- snmp-SnmpTableRow
- snmp-SnmpDevice
- snmp-SnmpDeviceFolder
- snmp-SnmpNetwork
- snmp-SnmpNumericProxyExt
- snmp-SnmpPointDeviceExt
- snmp-SnmpPointFolder
- snmp-SnmpPollScheduler
- snmp-SnmpRecipient
- snmp-SnmpStringProxyExt
- snmp-TrapTable
- snmp-TrapType

Component Reference Summary

Summary provides summary information on the components.

snmp-MIBListTable

MIBListTable captures information about MIB entries within SnmpDevices. The MIBListTable is available in the snmp module. Bajadoc is available at BMIBListTable.bajadoc

snmp-SnmpAgent

SnmpAgent represents the local Niagara station as an SNMP agent device holding agent data that can be viewed and changed via SNMP from an outside SNMP manager. The SnmpAgent is available in the snmp module. Bajadoc is available at BSnmpAgent.bajadoc

snmp-SnmpAgentBooleanProxyExt

SnmpAgentBooleanProxyExt contains information necessary to hold a boolean data value which can be set by an outside SNMP manager. Bajadoc is available at BSnmpAgentBoolean-ProxyExt.bajadoc

snmp-SnmpAgentNumericProxyExt

SnmpAgentNumericProxyExt contains information necessary to hold a float (or integer) data value which can be set by an outside SNMP manager. Bajadoc is available at BSnmpAgentNumer-icProxyExt.bajadoc

snmp-SnmpAgentPointDeviceExt

SnmpAgentPointDeviceExt is the container for SNMP agent proxy points representing SNMP agent data values. The SnmpAgentPointDeviceExt is available in the snmp module. Bajadoc is available at BSnmpAgentPointDeviceExt.bajadoc

snmp-SnmpAgentPointFolder

SnmpAgentPointFolder is the SNMP implementation of a folder under a SnmpAgent's Points extension. You add such folders using the `New Folder` button in the view of the Points extension. Each SnmpAgentPointFolder has its own view. The SnmpAgentPointFolder is also available in the snmp palette. Bajadoc is available at BSnmpAgentPointFolder.bajadoc

snmp-SnmpAgentStringProxyExt

SnmpAgentStringProxyExt contains information necessary to hold string data value which can be set by an outside SNMP manager. Bajadoc is available at BSnmpAgentStringProxyExt.bajadoc

snmp-SnmpAlarmDeviceExt

SnmpAlarmDeviceExt contains the standard generic Snmp trap-types. These are: coldStart, warmStart, linkDown, linkUp, authenticationFailure, egpNeighborLoss. See also, "SNMP device config-uration".

snmp-SnmpBooleanExport

The SnmpBooleanExport is a component that exports a boolean value from an SnmpNetwork Local Device to an SnmpNetwork Manager (Client). Add snmp export components under the SnmpExportTable or under folders that can be nested under the SnmpExportTable. Use the SnmpExportManager view to discover control points in your station and to add them to the SnmpEx-portTable. Also see, "About the Snmp Export Manager (SnmpExportTable)" on page 3-26.

snmp-SnmpBooleanProxyExt

SnmpBooleanProxyExt contains information necessary to read a boolean data value from an SNMP device. For numeric read SNMP data types, a value of zero is interpreted as a false, and anything else is interpreted as a true. For a read String SNMP data type, the string read will be compared with the true/false text for the point in order to determine the boolean value. The default is false (also used if cannot interpret).

Each read-only proxy point that represents a readable boolean SNMP data quantity will have an SnmpBooleanProxyExt to describe how to read the point. Bajadoc is available at BSnmpBoolean-ProxyExt.bajadoc

snmp-SnmpEnumExport

The SnmpEnumExport is a component that exports an enumerated value from an SnmpNetwork Local Device to an SnmpNetwork Manager (Client). Add snmp export components under the SnmpExportTable or under folders that can be nested under the SnmpExportTable. Use the SnmpExportManager view to discover control points in your station and to add them to the SnmpEx-portTable. Also see, "About the Snmp Export Manager (SnmpExportTable)" on page 3-26.

snmp-SnmpExportFolder

The SnmpEnumExport is a component that exports an enumerated value from an SnmpNetwork Local Device to an SnmpNetwork Manager (Client). Add snmp export components under the SnmpExportTable or under folders that can be nested under the SnmpExportTable. Use the SnmpExportManager view to discover control points in your station and to add them to the SnmpEx-portTable. Also see, "About the Snmp Export Manager (SnmpExportTable)" on page 3-26.

snmp-SnmpExportTable

SnmpExportTable is a container for the snmp export objects: SnmpBooleanExport, SnmpEnumExport, SnmpNumericExport, and SnmpStringExport. It can also contain nested folders for organizing export components. The default view of this component is the Snmp Export Manager view, where you can discover, add, and match to place export points in the SnmpExportTable. Also see, "About the Snmp Export Manager (SnmpExportTable)" on page 3-26.

snmp-SnmpNumericExport

The SnmpNumericExport is a component that exports a numeric value from an SnmpNetwork Local Device to an SnmpNetwork Manager (Client). Add snmp export components under the SnmpExportTable or under folders that can be nested under the SnmpExportTable. Use the SnmpExportManager view to discover control points in your station and to add them to the SnmpExportTable. Also see, "About the Snmp Export Manager (SnmpExportTable)" on page 3-26.

snmp-SnmpObjectDeviceExt

SnmpObjectDeviceExt is a container for any of the set of SnmpObject types. This includes read and read-write components for: SnmpObjectBooleanPoint, SnmpObjectEnumPoint, SnmpObjectNumericPoint, SnmpObjectStringPoint. The default view of this extension is the Snmp Object Manager view.

snmp-SnmpSequence

SnmpSequence objects contain an outputIndex, outputName, and outputValue slot (properties) that correspond to the columns in each SnmpTableRow. Each of these has the following editable properties: OID, Element Name, Element Type, Variable Type, and read-write option.

snmp-SnmpStringExport

The SnmpStringExport is a component that exports a string value from an SnmpNetwork Local Device to an SnmpNetwork Manager (Client). Add snmp export components under the SnmpExportTable or under folders that can be nested under the SnmpExportTable. Use the SnmpExportManager view to discover control points in your station and to add them to the SnmpExportTable. Also see, "About the Snmp Export Manager (SnmpExportTable)" on page 3-26.

snmp-SnmpTable

The SnmpTable component supports MIB tables. This object represents snmp data in a collection of BStatusValue objects that are organized in a sequence of rows, as they are "discovered" in a device. Each row consists of an object for each element, as represented in the MIB. The default view of the SnmpTable object is the Table Manager view, which presents the data in a table format. The row elements can be linked to other Niagara control objects. Read-only values can be linked to inputs and read-write values can be linked to inputs or outputs. Also see "About the Snmp Table Manager view" on page 3-24.

snmp-SnmpTableRow

.The SnmpTableRow contains an object for each element that is represented in the MIB table for the SnmpDevice. The row has an index, name, and value.

snmp-SnmpDevice

SnmpDevice represents a remote SNMP device that is treated as an agent. The SnmpDevice is available in the snmp module. Bajadoc is available at BSnmpDevice.bajadoc

snmp-SnmpDeviceFolder

SnmpDeviceFolder is the SNMP implementation of a folder under an SnmpNetwork. Typically, you add such folders using the **`New Folder`** button in the SnmpDeviceManager view of the SnmpNetwork. Each SnmpDeviceFolder has its own SnmpDeviceManager view. Bajadoc is available at BSnmpDeviceFolder.bajadoc

snmp-SnmpNetwork

SnmpNetwork represents a network of manageable SNMP devices. Can also be configured to handle sending and receiving SNMP trap messages, as well as respond to SNMP requests from outside managers. The SnmpNetwork is available in the snmp module. Bajadoc is available at BSnmpNetwork.bajadoc

snmp-SnmpNumericProxyExt

SnmpNumericProxyExt contains information necessary to read a float (or integer) data value from a SNMP device.

Each read-only proxy point that represents a readable float (integer) SNMP data quantity will have a SnmpNumericProxyExt to describe how to read the point. Bajadoc is available at BSnmpNumericProxyExt.bajadoc

snmp-SnmpPointDeviceExt

SnmpPointDeviceExt is the container for SNMP proxy points representing SNMP device data values. The SnmpPointDeviceExt is available in the snmp module. Bajadoc is available at BSnmpPointDeviceExt.bajadoc

snmp-SnmpPointFolder

SnmpPointFolder is the SNMP implementation of a folder under a SnmpDevice's Points extension. You add such folders using the **`New Folder`** button in the Snmp Point Manager view of the Snmp Point Device extension. Each SnmpPointFolder has its own view (Snmp Point Manager view). The SnmpPointFolder is also available in the snmp palette. Bajadoc is available at BSnmpPointFolder.bajadoc

snmp-SnmpPollScheduler

SnmpPollScheduler. The SnmpPollScheduler is available in the snmp module. Bajadoc is available at BSnmpPollScheduler.bajadoc

snmp-SnmpRecipient

SnmpRecipient recipient class is used to send alarm traps to an snmp network manager and to store alarms until acknowledged. The SnmpRecipient is available in the snmp module. Bajadoc is available at BSnmpRecipient.bajadoc

snmp-SnmpStringProxyExt

BSnmpStringProxyExt contains information necessary to read a String data value from a SNMP device.

Each read-only proxy point that represents a readable string SNMP data quantity will have an SnmpStringProxyExt to describe how to read the point. Bajadoc is available at BSnmpStringProxyExt.bajadoc

snmp-TrapTable

TrapTable captures information about stored trap types within an SnmpDevice. It is a frozen slot in an SnmpDevice (default name `Trap Types`), and contains child TrapType slots. The default view for an SnmpDevice's TrapTable is the SnmpTrapManager. Bajadoc is available at BTrapTable.bajadoc

snmp-TrapType

TrapType stores a possible trap type for an SnmpDevice, and resides under an SnmpDevice's TrapTable. Bajadoc is available at BTrapType.bajadoc