

Technical Document

Snmp V1/V2 Driver Guide

October 25, 2019

niagara⁴

Snmp V1/V2 Driver Guide

Tridium, Inc.

3951 Westerre Parkway, Suite 350
Richmond, Virginia 23233
U.S.A.

Confidentiality

The information contained in this document is confidential information of Tridium, Inc., a Delaware corporation ("Tridium"). Such information and the software described herein, is furnished under a license agreement and may be used only in accordance with that agreement.

The information contained in this document is provided solely for use by Tridium employees, licensees, and system owners; and, except as permitted under the below copyright notice, is not to be released to, or reproduced for, anyone else.

While every effort has been made to assure the accuracy of this document, Tridium is not responsible for damages of any kind, including without limitation consequential damages, arising from the application of the information contained herein. Information and specifications published here are current as of the date of this publication and are subject to change without notice. The latest product specifications can be found by contacting our corporate headquarters, Richmond, Virginia.

Trademark notice

BACnet and ASHRAE are registered trademarks of American Society of Heating, Refrigerating and Air-Conditioning Engineers. Microsoft, Excel, Internet Explorer, Windows, Windows Vista, Windows Server, and SQL Server are registered trademarks of Microsoft Corporation. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Mozilla and Firefox are trademarks of the Mozilla Foundation. Echelon, LON, LonMark, LonTalk, and LonWorks are registered trademarks of Echelon Corporation. Tridium, JACE, Niagara Framework, and Sedona Framework are registered trademarks, and Workbench are trademarks of Tridium Inc. All other product names and services mentioned in this publication that are known to be trademarks, registered trademarks, or service marks are the property of their respective owners.

Copyright and patent notice

This document may be copied by parties who are authorized to distribute Tridium products in connection with distribution of those products, subject to the contracts that authorize such distribution. It may not otherwise, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Tridium, Inc.

Copyright © 2019 Tridium, Inc. All rights reserved.

The product(s) described herein may be covered by one or more U.S. or foreign patents of Tridium.

Contents

About this Guide	5
Document Change Log	5
What's new in the Snmp Driver	5
Chapter 1 Snmp Driver Installation and Configuration	7
Configure the SnmpNetwork	7
Add an SnmpNetwork	8
Designing an Snmp network application	8
Create a MIB	11
Create Snmp proxy points	12
Chapter 2 Niagara Snmp Concepts	19
About Snmp network architecture	19
About nSnmp Palette components	20
Snmp network configuration	21
SnmpNetwork component properties	21
SnmpDevice configuration	25
About Snmp proxy points	26
Snmp client proxy points	27
Snmp agent proxy points	28
Snmp object proxy points	29
Snmp proxy point extensions	30
SnmpRecipient configuration	31
SnmpRecipient properties	32
About Snmp alarms (traps)	33
Generating Snmp traps	33
Receiving Snmp traps	35
Viewing Received Traps	36
About Snmp Alarm Data	37
Snmp alarm device extension properties	38
About Snmp manager	39
About MIBs	39
About locally generated MIBs	39
About the Snmp Servlet	41
Types of manager views	41
About the Snmp Point Manager view	42
About the Snmp Table Manager view	46
About the Snmp Trap Manager view	47
About the Snmp Agent Point Manager view	47
About the Snmp Export Manager (SnmpExportTable)	48
About the Snmp Object Manager view	49
Troubleshooting and debugging	50
Chapter 3 Snmp Plugins	53
Snmp Agent Point Manager	53
Snmp Device Manager	53

Snmp Export Manager.....	53
Snmp Object Manager.....	53
Snmp Point Manager	54
Snmp Table Manager	54
Snmp Trap Manager	54
Chapter 4 Snmp Components	55
MIB List Table	55
Snmp Agent.....	55
Snmp Agent Boolean Proxy Ext	56
Snmp Agent Numeric Proxy Ext.....	56
Snmp Agent Point Device Ext	56
Snmp Agent Point Folder	56
Snmp Agent String Proxy Ext.....	56
Snmp Alarm Device Ext	56
Snmp Boolean Export.....	56
Snmp Boolean Proxy Ext.....	56
Snmp Enum Export	57
Snmp Export Folder	57
Snmp Export Table.....	57
Snmp Numeric Export	57
Snmp Object Device Ext.....	57
Snmp Boolean Object Ext.....	57
Snmp Numeric Object Ext	57
Snmp String Object Ext	58
Snmp Enum Object Ext.....	58
Snmp Sequence	58
Snmp String Export	58
Snmp Table.....	58
Snmp Table Row	58
SnmpDevice.....	58
Snmp Device Folder	58
SnmpNetwork.....	59
Snmp Numeric Proxy Ext	59
Snmp Point Device Ext	59
Snmp Point Folder.....	59
N Poll Scheduler.....	59
Snmp Recipient	59
Snmp String Proxy Ext.....	59
Trap Table	60
snmp-TrapType	60

About this Guide

This topic contains important information about the purpose, content, context, and intended audience for this document.

Product Documentation

This document is part of the Niagara technical documentation library. Released versions of Niagara software include a complete collection of technical information that is provided in both online help and PDF format. The information in this document is written primarily for Systems Integrators. In order to make the most of the information in this book, readers should have some training or previous experience with Niagara 4 software, as well as experience working with JACE network controllers.

Document Content

This document describes installation and configuration of the Snmp version 1 and version 2 driver. It also includes basic Niagara Snmp concept and task descriptions with short descriptions for most of the components and views included in the Snmp module.

NOTE: This document is updated to include any changes resulting from redevelopment of the Snmp driver using the Niagara nDriver framework.

CAUTION: Protect against unauthorized access by restricting physical access to the computers and devices that manage your building model. Set up user authentication with strong passwords, and secure components by controlling permissions. Failure to observe these recommended precautions could expose your network systems to unauthorized access and tampering.

Document Change Log

Updates (changes/additions) to this document are listed below.

October 25, 2019

In the topic, "About this guide", added a caution note alerting customers to restrict access to all computers, devices, field buses, components, etc., that manage their building model.

November 2, 2017

Minor correction in the topic, "About Snmp alarms (traps)".

June 6, 2016

Initial publication.

What's new in the Snmp Driver

The following list describes the major changes in the Snmp Driver.

- **New Snmp module**
A new "nSnmp" module is added.
- **Snmp Recipient Property**
The Network Manager Snmp version property is added in the Snmp Recipient.
- **Snmp point manager Discovered pane**
The new column DisplayHint is added to the **Discovered** pane of the Snmp point manager.
- **Snmp point manager Database pane**
The new column Fault Cause added to the **Database** pane of the Snmp point manager.
- **SnmpNetwork component properties**
The Receive Config appearance changed.

The Trap Config appearance changed.

A new field "Default Network Manager Snmp Version" is added.

- **Configure the Snmp service on Windows**

New topic "Configure the Snmp service on Windows" is added.

- **TagIt button**

This button is added in all manager views, for future use, to allow tagging components from manager views. It is not active in this initial version.

- **Troubleshooting and debugging**

The logging is changed from Tridium Log framework to java.util.logging.

Chapter 1 Snmp Driver Installation and Configuration

Topics covered in this chapter

- ◆ Configure the SnmpNetwork

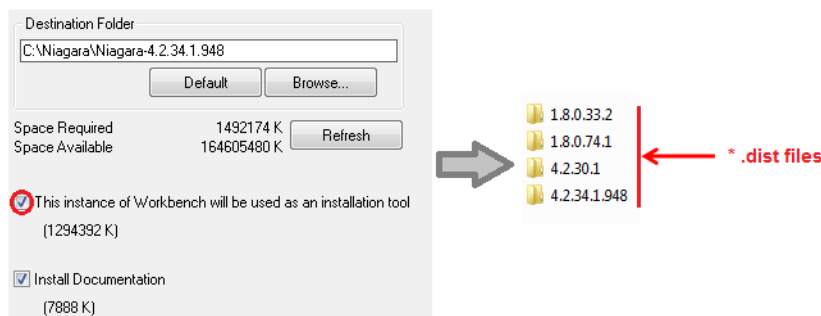
The following list describes the items that you need to consider for installation and basic configuration of the Snmp Driver.

Licensing

To use the 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 4.2.nn (or later) that was installed with the “installation tool” option selected, as shown in the following figure. 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 installation directory under a “sw” subdirectory. For details, see “About your software database” in the Niagara Platform Guide.



Snmp module

Apart from installing the 4.2.nnn (or newer) version 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 *Niagara Platform Guide*.

The remote controller is now ready for Snmp Network configuration in its running station, as described in the “Configure the SnmpNetwork” topic.

NOTE: Basic procedures for using the online features of the driver, including discovery of online Snmp devices and points, are discussed in [Create Snmp Agent proxy points, page 12](#) and [Create Snmp proxy points, page 12](#).


Configure the SnmpNetwork

This section provides procedures and descriptions that are commonly required and used with the **Snmp Driver** in typical online scenarios.

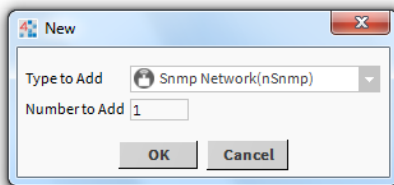
To configure the SnmpNetwork, first add the SnmpNetwork and then design the SnmpNetwork application.

Add an SnmpNetwork

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

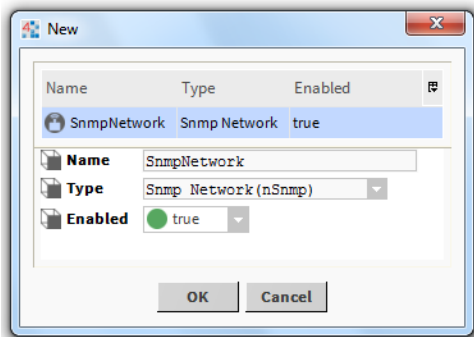
When you add an SnmpNetwork to a station, a **Local Device** () component comes with the SnmpNetwork component. This **Local Device** (also referred to as an “Snmp Agent”) allows you to expose data to remote Snmp sources (managers). Only one local device is allowed under the SnmpNetwork and is pre-configured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

- Step 1 Double-click the station **Drivers** container. 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 *Niagara Drivers Guide*.
- Step 3 Select **SnmpNetwork (nSnmp)**, number to add: 1 and click **OK**.



NOTE: You can add only one SnmpNetwork (nSnmp) object to a station.

- Step 4 This brings up the **New** dialog box. Type a name in the **Name** field (or accept the default name) and select **Enabled** to enable the network.



- Step 5 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 **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 “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 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 Snmp Agent object may be added to an SnmpNetwork.

Snmp Manager

Set up and use this device for creating a “client” type application that you use to manage one or more Snmp Agent devices on your network. Under the Snmp Device 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 SnmpNetwork object, a **Local Device** component comes with it.

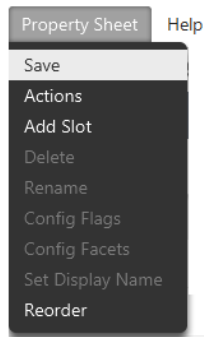
When you add an SnmpNetwork object, a **Local Device** component comes with it.

- 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, tick the option to `true`.
- 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.

SnmpNetwork Actions & Topics

Display Name	Value	Commands
Status	{ok}	
Enabled	<input checked="" type="checkbox"/> true	
Fault Cause		
Health	Ok [15-Feb-16 4:42 PM IST]	
Alarm Source Info	Alarm Source Info	
Monitor	Ping Monitor	○
Tuning Policies	Tuning Policy Map	
Poll Scheduler	N Poll Scheduler	○
Enterprise	1.3.6.1.4.1.4131	
Contact	contact	
System Name	name	
Location	location	
Snmp Receive Requests	<input checked="" type="checkbox"/> Enabled (Can send traps)	

- Step 4 Right-click on the **Property Sheet** from the **Menu** bar and select the option **Save** from the list.

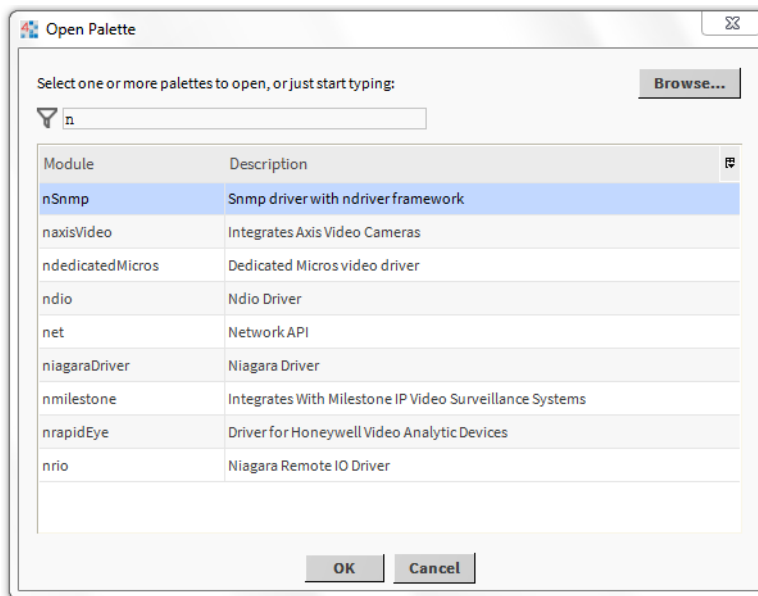


You are now ready to add **Snmp Agent Points** under the **Local Device** in order to expose the data to Snmp requests from outside sources.

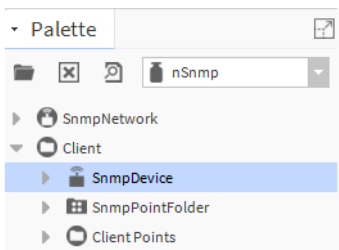
Set up an Snmp manager application

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

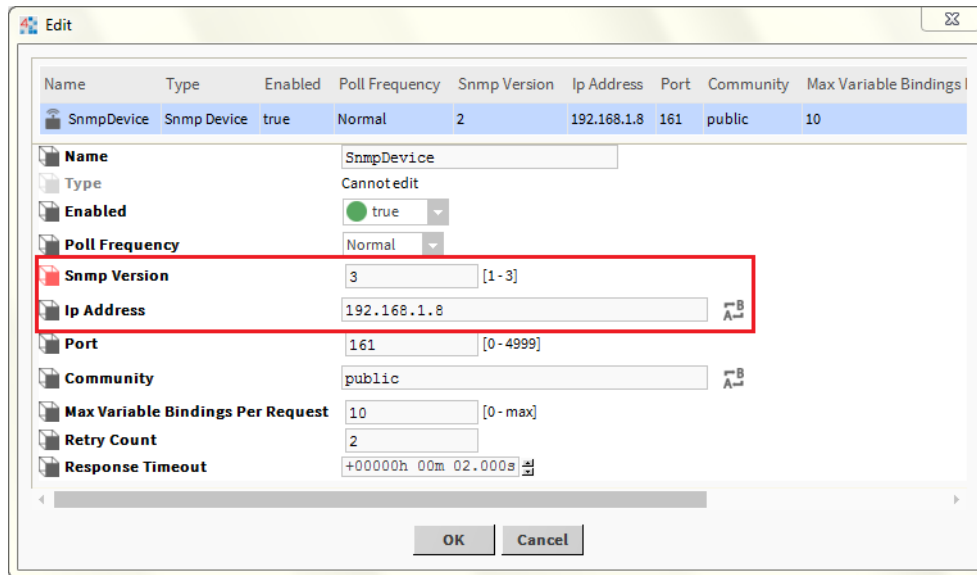
- Step 1** Under the **Drivers** node of your station, double click on the SnmpNetwork node in the nav side bar. The **Snmp Device Manager** view displays.
- Step 2** Using the **Palette** side bar controls, open the local **nSnmp Module** palette.



- Step 3** 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 4 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 5 Double click on the added device in the **Snmp Device Manager** view. The **Edit** dialog appears.
- Step 6 In the **Edit** dialog box, set the **Snmp Version** and **Ip Address** of the JACE (Agent), then click the **OK** button. The **SnmpDevice** is added under the **SnmpNetwork**.

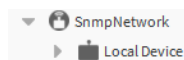


Create a MIB

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**. This procedure describes how to use the **CreateMIB** feature on an **Snmp Local Device**.

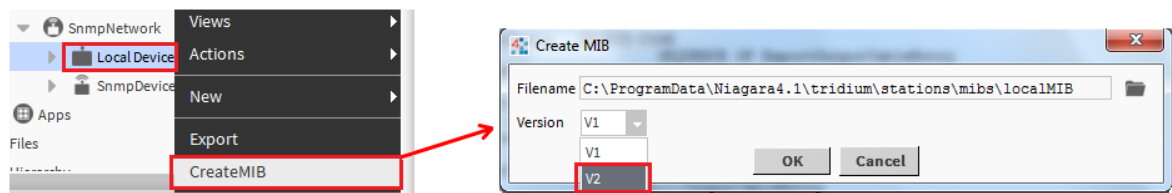
- Step 1 In the **Nav** tree, expand your station tree to expose the **Local Device** (**Station**→ **Config**→ **Drivers**→ **SnmpNetwork**).

The **Local Device** displays under the **SnmpNetwork** node.



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

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



- Step 3 In the **Filename**, type the path or browse to a location to specify where you want to save the MIB file and select the **Version** to V2 or V1.

The MIB file displays in the MIB editor.

- Step 4 Click the **OK** button.

- Step 5 If you need to make changes to the MIB file, use the **AX Text File Editor** to edit the text and save the MIB file when you are finished.

The file is saved and the **AX Text File Editor** closes. The custom MIB file is now available for using with "Discovery" and value updates ("Walking the MIB").

The connection to the desired Snmp licensed station with the SnmpNetwork must be properly configured. For more information, refer to [About locally generated MIBs, page 39](#).

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 SnmpNetwork, 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**.

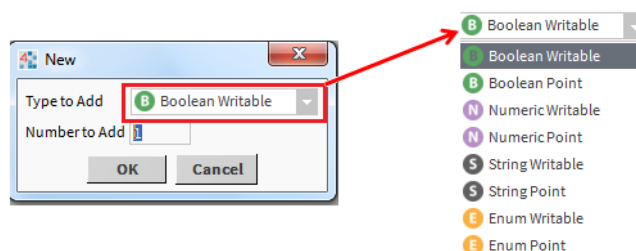
For general information about using the **Point Manager** view, refer to "About the Point Manager" in the *Niagara Drivers Guide*.

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 **nSnmp** 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.

Create Snmp Agent proxy points

To create Snmp Agent proxy points in a device, you must first have a properly configured SnmpNetwork with a **Local Device** under it. Snmp Agent proxy points are typical point types with a special Snmp Agent extension included.

- Step 1 In the **Nav** side bar, under the station SnmpNetwork 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. 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 station. OID values are automatically assigned for each point.

Name	Type	Index	Default Value	Enabled	Device Facets	Facets
B BooleanWritable	Boolean Writable	1		true	trueText=true,falseText=false	trueText=true,falseText=false

Name	BooleanWritable
Type	Boolean Writable
Index	1
Default Value	Cannot edit
Enabled	<input checked="" type="radio"/> true
Device Facets	trueText=true,falseText=false >> ⌚
Facets	trueText=true,falseText=false >> ⌚
Conversion	Default
Tuning Policy Name	Default Policy

OK Cancel

Note the following about entries in the **New** 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 the Niagara control point type to use for the proxy point.
NOTE: Unlike other editable entries of the **New** dialog box, you cannot edit **Type** later.
- Index**
 As Snmp Agent proxy points are added to the **Local 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→Set** 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 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 Agent Point 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

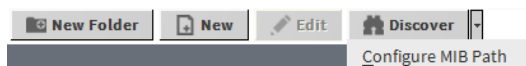
Snmp Client proxy points are typical point types with a special Snmp Client extension included.

Prerequisites:

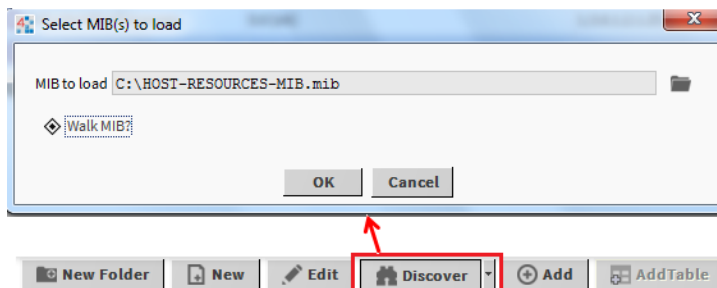
To create **Snmp Client proxy** points in a device, you must first have a properly configured SnmpNetwork with an **SnmpDevice** under it and a valid MIB file that is accessible from your Workbench application.

Step 1 In the **Nav** side bar, under the station **SnmpNetwork** node, expand the **SnmpDevice** 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.



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.

Name	Type	ObjectIdentifier	Variable Type	Enabled	Device Facets	Facets
hrDeviceIndex_2	Numeric Point	1.3.6.1.2.1.25.3.2.1.1.1	Integer Type	true	units=null,precision=1,min=-inf,max=+inf	units=...

Name: hrDeviceIndex_2
Type: Numeric Point
ObjectIdentifier: 1.3.6.1.2.1.25.3.2.1.1.1
Variable Type: Integer Type
Enabled: true
Device Facets: units=null,precision=1,min=-inf,max=+inf
Facets: units=null,precision=1,min=-inf,max=+inf
Conversion: Default
Tuning Policy Name: Default Policy

OK Cancel

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 *Niagara 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 the 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 **SnmpDevice**, 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 **SnmpDevice** 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 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

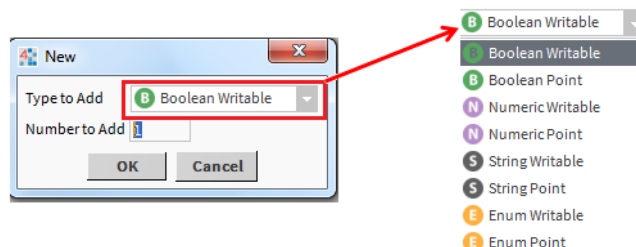
Snmp Client proxy points are typical point types with a special Snmp Client extension included. This topic gives the information to create the Snmp Client proxy points manually in a device.

Step 1 In the **Nav** side bar, under the SnmpNetwork node, expand the **SnmpDevice** 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, 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 station. OID values are not automatically assigned for the added point because the point has not been “discovered”.

Name	Type	ObjectIdentifier	Variable Type	Enabled	Device Facets	Facets
NumericPoint	Numeric Point	1.3.	Integer Type	true	units=null,precision=1,min=-inf,max=+inf	units=null,pr

Name NumericPoint
Type Numeric Point
ObjectIdentifier 1.3.
Variable Type Integer Type
Enabled true
Device Facets units=null,precision=1,min=-inf,max=+inf
Facets units=null,precision=1,min=-inf,max=+inf
Conversion Default
Tuning Policy Name Default Policy

OK Cancel

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 the Niagara control point type to use for the proxy point.
NOTE: Unlike other editable entries in the **Add** dialog, you cannot edit **Type** property 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 **SnmpDevice** 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 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 2 Niagara Snmp Concepts

Topics covered in this chapter

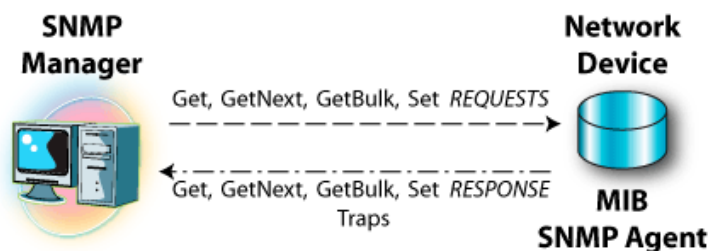
- ◆ About Snmp network architecture
- ◆ About nSnmp Palette components
- ◆ Snmp network configuration
- ◆ SnmpDevice configuration
- ◆ About Snmp proxy points
- ◆ SnmpRecipient configuration
- ◆ About Snmp alarms (traps)
- ◆ About Snmp manager
- ◆ About MIBs
- ◆ About locally generated MIBs
- ◆ Types of manager views
- ◆ Troubleshooting and debugging

This section includes topics that describe Snmp concepts that comprise the Snmp driver.

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 Snmp Agent responds to requests for information and actions from the Snmp Manager. The Snmp 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



Communication between the Snmp Agent and Manager occurs in one of the following forms:

- Get, GetBulk, and GetNext requests

The Snmp Manager requests information from the Snmp Agent; the Snmp Agent returns the information in a Get response message.

- Set requests

The Snmp Manager changes the value of a MIB object controlled by the Snmp Agent; the Snmp Agent indicates status in a Set response message.

- Traps notification

The Snmp Agent sends traps to notify the Manager of significant events that occur on the network device.

The Snmp driver uses the standard Niagara network architecture. See "About Network architecture" in the *Niagara 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 Snmp Manager and 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 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 SnmpDevice. 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

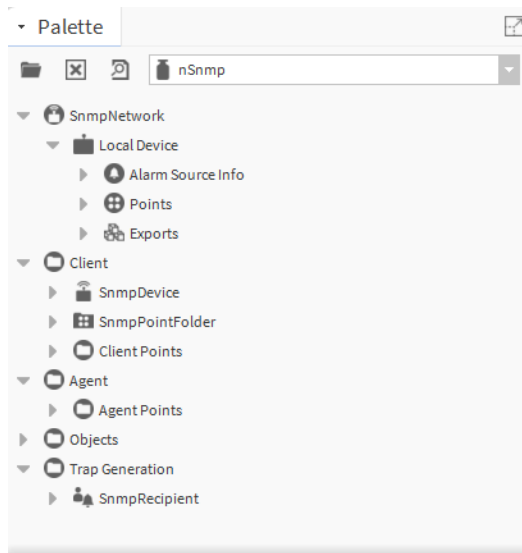
An agent is an Snmp application that responds to queries from Snmp Manager applications. The Snmp Agent is responsible for retrieving and updating local management information based on the requests of the Snmp Manager. The Snmp Agent also notifies registered managers when significant events or traps occur.

The 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 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 nSnmp Palette components

Open the **nSnmp Palette** in Workbench to view components used in an SnmpNetwork.

Following is a summary of the **Palette** contents:



SnmpNetwork

This component contains the required **Local Device** component and its child components:

- **Alarm Source Info**
- **Points**
- **Exports.**

Client

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**
- **Points**
- **Traps.**

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

The agent components include **Agent Points** that are standard point types with proxy extensions pre-configured for an **SnmPDevice**.

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 **Local Device**.

Trap Generation

This folder contains the **SnmPRecipient** component.

SnmP network configuration

These topics describe how to configure an SnmpNetwork

SnmP devices communicate using the Snmp protocol over a network. The SnmpNetwork is designed to handle multiple **SnmPDevices** using the Snmp version 1 or 2 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.

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 *Niagara Drivers Guide*.

▶ Poll Scheduler	N Poll Scheduler	
Enterprise	1.3.6.1.4.1.4131	
Contact	contact	
System Name	name	
Location	location	
Snmp Receive Requests	<input checked="" type="checkbox"/> Enabled (Can send traps)	
▶ Receive Config	local:0 tcp	
Ignore Requests From Unrecognized Sources	<input type="checkbox"/> false	
Recognized Sources	Network Manager List	
Check Community On Requests	<input checked="" type="checkbox"/> true	
Read Only Community	public	
Read Write Community	public	
Snmp Receive Traps	<input checked="" type="checkbox"/> true	
▶ Trap Config	local:0 tcp	
Default Network Manager Ip Address	localhost	
Default Network Manager Traps Port	162	
Default Network Manager Snmp Version	1	
Default Network Manager Traps Community	public	
Snmp Alarm Table Capacity	500	

N Poll Scheduler

Used to configure device-level polling. See the *Niagara Drivers Guide* for more information about polling. for details on configuring the Niagara Poll Scheduler.

Enterprise

Displays the enterprise OID (Object Identifier) 1.3.6.1.4.1.4131 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 Snmp V1 or Snmp V2 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.

Receive Config

- **Fault Cause**

It gives a description if the point goes into the fault.

- **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.

Receive Config	local:161 udp
Fault Cause	Address already in use: Cannot bind
Receive Port	161 udp
Public Server Port	161
Ip Protocol	Udp

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:

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.

Traps Config

- **Fault Cause**

It gives a description if the point goes into the fault.

- **Receive Port**

Specifies the port that the station uses to receive Snmp trap messages from external Snmp devices. The default is port 162.

Trap Config	
Fault Cause	
Receive Port	162 udp
Public Server Port	162
Ip Protocol	Udp

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 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 Snmp Version

Specifies the Snmp version whether it is Snmp V1 or Snmp V2.

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".

SnmpDevice 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 **SnmpDevice**. The **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**.

Property Sheet	
SnmpDevice (Snmp Device)	
Status	{ok}
Enabled	<input checked="" type="checkbox"/> true
Fault Cause	
Health	Ok [14-Mar-16 12:30 PM IST]
Alarm Source Info	Alarm Source Info
Poll Frequency	Normal
Points	Snmp Point Device Ext
Traps	Snmp Alarm Device Ext
Snmp Version	2 [1-2]
Retry Count	2
Response Timeout	+00000h 00m 02.000s
Max Variable Bindings Per Request	10 [0-max]
Ip Address	localhost
Port	161 [0-4999]
Community	public
Mib	C:\HOST-RESOURCES-MIB.mib

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 *Niagara 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.

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 **SnmpDevice** 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, page 38](#).

Snmp Version

Specifies the version of the Snmp protocol to use for communication with this **SnmpDevice**. The default is 2 (representing Snmp v2), and the only other option is 1 (representing Snmp v1).

Retry Count

Specifies the number of times any individual Snmp request made to this **SnmpDevice** will be retried when receiving a null response before considering the request to be a communication failure.

Response TimeOut

Specifies the maximum amount of time to wait for a response after sending an Snmp request to this **SnmpDevice**. 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.

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**.

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.

Community

Specifies the community string field to use for outgoing Snmp request messages sent to the **SnmpDevice**. The default value is "public".

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, page 27](#) 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.

Snmp agent 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 **SnmpDevice** (usually an outside Snmp manager). When Snmp agent proxy points are added to an Snmp agent 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 Snmp v1 and Snmp v2 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. The data contained in the Output Table will not accept SET requests as it contains read-only data. As Snmp Agent 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, page 28](#) for more information.

Snmp agent (server) export points

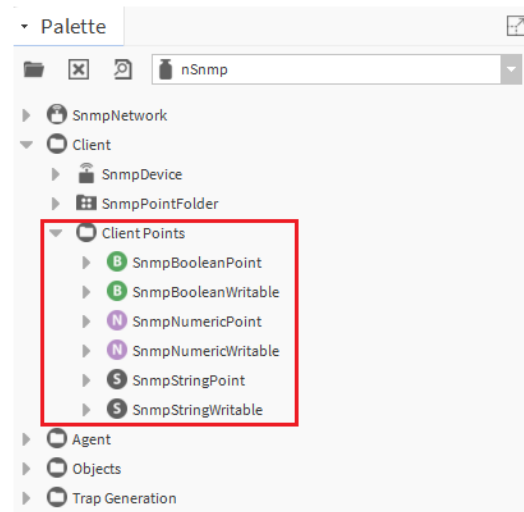
These types of points appear under the Snmp Export Table, which has the **Snmp Export Manager** as its default view. You can use the **Snmp Export Manager** view to discover and add control points from a **Local Device** to the Export Table. The discovery process uses a **Bql Query** dialog box to help 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 control points with unique Snmp proxy point extensions that define the Snmp-specific behavior of the point.

Refer to *Niagara 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, page 42](#) for more information about the **Point Manager** view. The client proxy points and extensions are also available in the **Client** folder of the **nSnmp Palette**.



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.

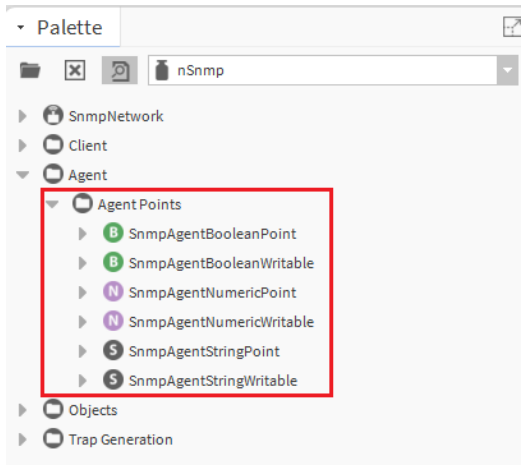
These Snmp proxy points support reading or writing the following Snmp variable types:

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

Snmp agent proxy points

Snmp agent proxy points must be located under an Snmp **Point** device extension. Snmp agent proxy points consist of standard control points with unique Snmp agent proxy point extensions which define the Snmp agent-specific behavior of the point.

The Snmp agent proxy points leverage off the standard control point types. Refer to the appropriate *Niagara Drivers Guide* documentation for details on the basic control point types. The Snmp agent proxy points are available in the **Agent Points** folder of the **nSnmp Palette**.



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.

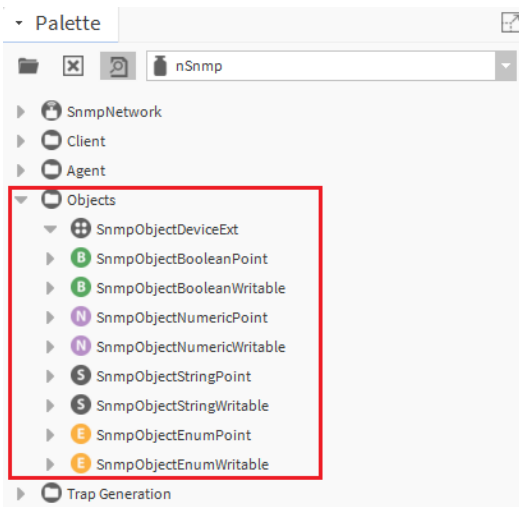
SnmpAgentStringWritable

These control point types host the following read-only SnmpAgent proxy point extensions that populate the Output Table: SnmpAgentStringWritableProxyExt.

Snmp object proxy points

Snmp agent proxy points must be located under an Snmp **Point** device extension. Snmp agent proxy points consist of standard control points with unique Snmp agent proxy point extensions which define the SnmpAgent-specific behavior of the point.

Refer to *Niagara Drivers Guide* documentation for details on the basic control point types themselves. The object proxy points and extensions are available in the **Object** folder of the **nSnmp Palette**.



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
- SnmpObjectNumericWritable
- SnmpObjectStringPoint
- SnmpObjectStringWritable
- SnmpObjectEnumPoint
- SnmpObjectEnumWritable

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 *Niagara 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.

Object Identifier	<input type="text"/>
Variable Type	Integer Type
Description	<input type="text"/>
Object Identifier	<input type="text"/>
Index	2
Default Value	<input checked="" type="checkbox"/> autoSet <input type="text"/>

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 **Snmp Point Manager** (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.

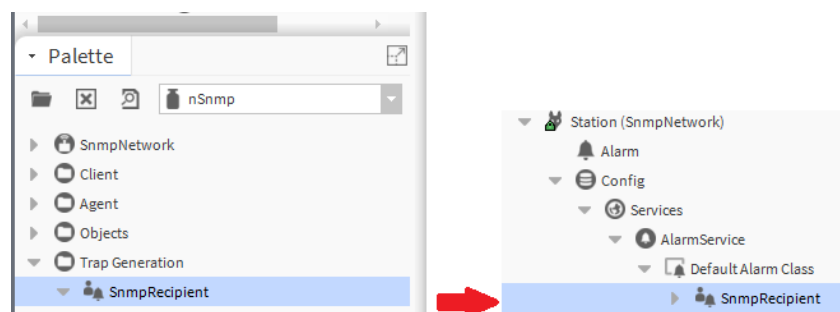
Default Value (Agent)

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 user can specify a permanent default value (any external Snmp SET request will not change the default value). 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 AlarmRecipients. They function the same as the standard AlarmRecipient component 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→AlarmService**.



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

Notice that the **Alarm** slot of the Alarm Class component is linked to the **Route Alarm** slot of the **SnmpRecipient** component. This ensures that 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"](#), page 32. 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 alarms (in the form of Snmp v1 traps) to a defined Snmp manager whenever a new alarm record is routed to the **SnmpRecipient**.

Receiving Snmp traps

Use the **SnmpRecipient** to store any received alarm records in the **Snmp Alarm Table** of the SnmpNetwork. The **SnmpRecipient** receives the alarm record from the linked **Default Alarm Class** and packages it into an Snmp v1 trap message to send to the network manager.

SnmpRecipient properties

SnmpRecipient properties are configurable from the **SnmpRecipient** component **AX Property Sheet** view.

By selecting the **AX Property Sheet** view of an **SnmpRecipient** component, you can view and configure the **SnmpRecipient** properties.

The screenshot shows the 'SnmpRecipient (Snmp Recipient)' AX Property Sheet. The 'Route Acks' property is set to 'true'. The 'Route Alarms To Network Manager As Traps' and 'Use SnmpNetwork's Default Network Manager' properties are checked. The 'Network Manager Config' section includes fields for 'Network Manager Ip Address', 'Network Manager Traps Port' (162), 'Network Manager Snmp version' (1), and 'Network Manager Traps Community' (public). The 'Snmp Alarm Table' property is set to 'Store Received Alarms'.

The **Time Range**, **Days of Week**, and **Transitions** properties are standard to all Alarm Recipients and detailed in the *Niagara Drivers Guide*. The Snmp-specific properties are described in the following section.

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 version 1 trap messages to the specified Snmp network manager whenever a new alarm record is received. If this option is not selected, then this service is not enabled and any received 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, 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 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 Snmp version

This property specifies the Snmp version.

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 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 objects, all alarms that generate trap messages (or all received Snmp trap messages that generate alarms) can be viewed in a alarm console. In order to generate and receive Snmp traps, you must properly configure the **SnmpNetwork** and **SnmpRecipient** components.

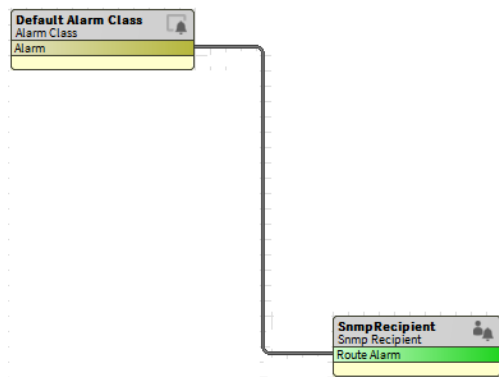
For additional information about alarming and the alarm console, see the *Niagara Alarms Guide*.

Generating Snmp traps

You can generate and send alarms (in the form of Snmp v1 traps) to a defined Snmp manager whenever a new 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.

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 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 **nSnmp Palette**. You can add as many **SnmpRecipients** as you need to configure your Snmp integration. An **SnmpRecipient** component linked to the **Default Alarm Class**. In this example, any 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.



NOTE: All Snmp traps generated and sent by the **SnmpRecipient** are Snmp v1 messages. The Snmp driver currently does not support generating and sending Snmp v2 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.

- Step 1 Open the **nSnmp Palette** in the **Palette** side bar.
- Step 2 Under the **Trap Generation** folder, drag and drop an **SnmpRecipient** component onto the **Alarm-Service** property sheet or onto the **AlarmService** component in the **Nav** side bar.
- Step 3 Select the **Property Sheet** view of the **SnmpRecipient** and set the Time Range, Days of Week, and Transitions properties as desired. Refer to [SnmpRecipient properties, page 32](#).
- Step 4 On the **SnmpRecipient** property sheet, set the **Route Acks** property to **true**.

SnmpRecipient Actions & Topics Slot Details

Display Name	Value	Commands
Time Range	12 : 00 : 00 AM 12 : 00 : 00 PM	
Days Of Week	<input checked="" type="checkbox"/> Sun <input checked="" type="checkbox"/> Mon <input checked="" type="checkbox"/> Tue <input checked="" type="checkbox"/> Wed <input checked="" type="checkbox"/> Thu <input checked="" type="checkbox"/> Fri <input checked="" type="checkbox"/> Sat	
Transitions	<input checked="" type="checkbox"/> toOffnormal <input checked="" type="checkbox"/> toFault <input checked="" type="checkbox"/> toNormal <input checked="" type="checkbox"/> toAlert	
Route Acks	<input checked="" type="checkbox"/> true	
Network Manager Config	Network Manager Configuration	
Route Alarms To Network Manager	<input checked="" type="checkbox"/> true	
Use Default Network Manager	<input checked="" type="checkbox"/> true	
Network Manager Ip Address	localhost	
Network Manager Traps Port	162	
Network Manager Snmp Version	1	
Network Manager Traps Community	public	
Snmp Alarm Table	<input checked="" type="checkbox"/> Store Received Alarms	

- 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.

- **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.

- **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".

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

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. You also need to configure properties that specify whether or not you want to receive traps from unrecognized sources.

 Snmp Receive Traps	<input checked="" type="checkbox"/> true
 Trap Config	local:0 tcp
 Default Network Manager Ip Address	<input type="text"/>
 Default Network Manager Traps Port	<input type="text" value="162"/>

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 SnmpNetwork established and running in your station. You also need to have an **SnmpDevice** component in your network application for any devices that may potentially send out trap messages.

- 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 **Receive Port** property to the desired port number. (Port 162 is the default value).
- Step 4 On the **SnmpNetwork** property sheet, click the **Save** button from the **Menu** bar. 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 **SnmpDevice**, 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. The Snmp trap generation is configured and enabled.

Viewing Received Traps

Traps information may be viewed in the alarm console.

Traps information may be viewed in the alarm console. Route traps to the alarm console by linking the **ConsoleRecipient** to the appropriate **AlarmClass** component.



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. The **Alarm Record** dialog box provides a full description of the message.

The screenshot shows the 'Open Alarm Sources' window with a table of alarm sources. A red box highlights the first row, which is selected. A red arrow points from this row to the 'Alarm Record' window below.

Timestamp	Source	Alarm Data
26-Feb-16 2:55:05 PM IST	SnmpNetwork SnmpDevice	msgText=Ping Success,sourceName=SnmpNetwork SnmpDevice,TimeZone=Asia/Calcutta (+5:30)
11-Feb-16 11:24:53 AM IST	SnmpNetwork SnmpDevice1	msgText=Ping failed,sourceName=SnmpNetwork SnmpDevice1,TimeZone=Asia/Calcutta (+5:30)

The 'Alarm Record' window displays the following details for the selected alarm:

Timestamp	26-Feb-16 2:55:05 PM IST
Uuid	babed325-2bd0-4d07-957f-7257476b1b48
Source State	Normal
Ack State	Unacked
Ack Required	true
Source	SnmpNetwork SnmpDevice local: station: slot:/Drivers/SnmpNetwork/SnmpDevice
Alarm Class	Default Alarm Class
Priority	255
Normal Time	10-Mar-16 11:50:21 AM IST
Ack Time	null
User	Unknown User
Alarm Data	Escalated Message Text Ping Success Notes >> Source Name SnmpNetwork SnmpDevice Time Zone Asia/Calcutta (+5:30)
Alarm Transition	Offnormal
Last Update	10-Mar-16 11:50:21 AM IST

At the bottom of the 'Alarm Record' window are buttons: Acknowledge, Hyperlink, Notes, and Close.

For complete details about using the alarm console, including information about adding or removing alarm console columns, refer to “About the alarm console”.

About Snmp Alarm Data

The **Alarm Data** field of the **Alarm Record** displays a received trap message using a `Trap_Data` facet in one of the following forms, depending on Snmp version.

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 '.

Alarm Data	
Agent_addr	10.78.64.158
Enterprise	1.3.6.1.4.1.4131.1
Escalated	
Generic_trap	6
Notes	>>
Specific_trap	1
Time_stamp	420152(01-Jan-70 6:40 AM IST)
Time Zone	Asia/Calcutta (+5:30)
Trap_Data	Received unrecognized v1 Trap from SnmpDevice66(/10.78.64.158)
Variable_bindings_0	12-Apr-16 12:07 PM IST (OID: 1.3.6.1.4.1.4131.1.3.1.1)
Variable_bindings_1	4f3bc66d-aac5-4fd9-ab2a-1cf34f78db7a (OID: 1.3.6.1.4.1.4131.1.3.1.2)
Variable_bindings_10	Unknown User (OID: 1.3.6.1.4.1.4131.1.3.1.11)
Variable_bindings_11	presentValue=28.7; fromState=normal; toState=fault; alarmValue=28.70; lowLimit=30.0; faultValue=28.7; msgText=Alarm from 10.78.64.158..Value Below 30!!; deadband=0.0; sourceName=Alrm_2; Count=53; highLimit=40.0; status={ok} @ 10; TimeZone=Asia/Calcutta (+5:30); escalated= (OID: 1.3.6.1.4.1.4131.1.3.1.12)
Variable_bindings_12	2 (OID: 1.3.6.1.4.1.4131.1.3.1.13)
Variable_bindings_13	12-Apr-16 12:07 PM IST (OID: 1.3.6.1.4.1.4131.1.3.1.14)
Variable_bindings_2	2 (OID: 1.3.6.1.4.1.4131.1.3.1.3)
Variable_bindings_3	1 (OID: 1.3.6.1.4.1.4131.1.3.1.4)

Snmp v1 trap message

For a recognized Snmp v1 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]]"
```

Snmp v2 trap message

For a recognized Snmp v2 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 *Niagara 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.

Last Received Time

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 Received Trap

This StringElement output displays the detailed message of the last received Snmp trap message for this **SnmpDevice** since station startup.

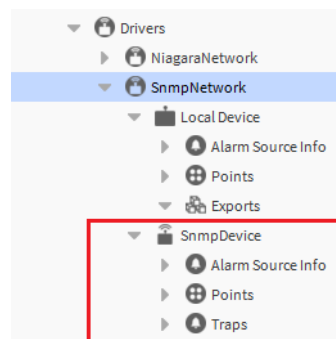
Ignore Unrecognized Traps

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 SnmpNetwork as an Snmp manager to view and control points on a remote SnmpNetwork station.

The remote station must contain an **SnmpDevice** component that is configured as an Snmp agent. The Snmp agent exposes data through the use of an SnmpAgent device and SnmpAgent proxy points. 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 alarms via Snmp requests.



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.

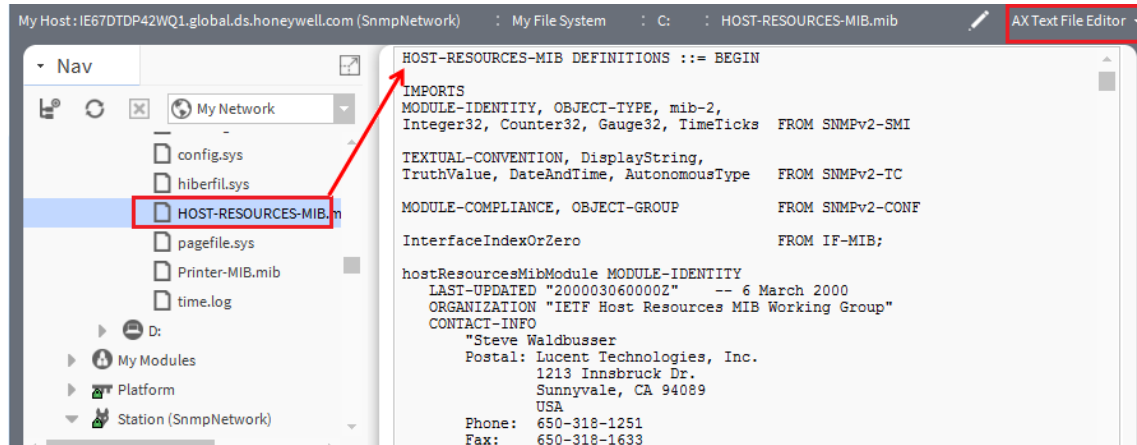
The name space for MIB object identifiers is hierarchical. It is structured so that each manageable object can be assigned a globally unique name. Authority for parts of the name space is assigned to individual organizations. This allows organizations to assign names without consulting an Internet authority 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

Create a MIB from the popup menu on an SnmpNetwork **Local Device** (Refer to [Create a MIB, page 11](#)) or by using the Snmp Servlet through a browser connection to your station (Refer to [Create a MIB using the Snmp Servlet, page 41](#)).

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 below. After compiling the MIB, an Snmp manager application can use the information to read and write data to and from the station.



“Standard” MIBs are included in the nSnmp 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 below. 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. Refer to [About the Snmp Point Manager view, page 42](#) for more information about the **Snmp Point Manager** view.

Name	OID	Type	Value	Description
hrSystemUptime	1.3.6.1.2.1.25.1.1.0	Timeticks		The amount of time since this host was last initialized. Note that this is different from sysUpTime in the SNMPv2-MIB [RFC1907] because sysUpTime is the time since the system was last rebooted.
hrSystemDate	1.3.6.1.2.1.25.1.2.0	String Type		The host's notion of the local date and time of day.
hrSystemInitialLoadDevice	1.3.6.1.2.1.25.1.3.0	Integer Type		The index of the hrDeviceEntry for the device from which this host is configured to load its initial operating system configuration (i.e., which device).
hrSystemInitialLoadParameters	1.3.6.1.2.1.25.1.4.0	String Type		This object contains the parameters (e.g., a pathname and parameter) supplied to the load device when requesting the initial operating system configuration.
hrSystemNumUsers	1.3.6.1.2.1.25.1.5.0	Gauge		The number of user sessions for which this host is storing state information. A session is a collection of processes requiring a single act of login.
hrSystemProcesses	1.3.6.1.2.1.25.1.6.0	Gauge		The number of process contexts currently loaded or running on this system.
hrSystemMaxProcesses	1.3.6.1.2.1.25.1.7.0	Integer Type		The maximum number of process contexts this system can support. If there is no fixed maximum, the value should be zero. On systems that support dynamic loading, the value should be the maximum number of contexts that can be loaded at any one time.
hrMemorySize	1.3.6.1.2.1.25.2.2.0	Integer Type		The amount of physical read-write main memory, typically RAM, contained by the host.
hrStorageTable	1.3.6.1.2.1.25.2.3			The (conceptual) table of logical storage areas on the host. An entry shall be placed in the storage table for each logical area of storage that is supported by the host.
hrStorageEntry	1.3.6.1.2.1.25.2.3.1			A (conceptual) entry for one logical storage area on the host. As an example, an instance of the hrStorageType object might be named hrStorage1.
hrStorageIndex	1.3.6.1.2.1.25.2.3.1.1	Integer Type		A unique value for each logical storage area contained by the host.
hrStorageType	1.3.6.1.2.1.25.2.3.1.2	OID		The type of storage represented by this entry.
hrStorageDescr	1.3.6.1.2.1.25.2.3.1.3	String Type		A description of the type and instance of the storage described by this entry.
hrStorageAllocationUnits	1.3.6.1.2.1.25.2.3.1.4	Integer Type		The size, in bytes, of the data objects allocated from this pool. If this entry is monitoring sectors, blocks, buffers, or packets, for example, the value should be the size of the smallest object that can be allocated from this pool.
hrStorageSize	1.3.6.1.2.1.25.2.3.1.5	Integer Type		The size of the storage represented by this entry, in units of hrStorageAllocationUnits. This object is writable to allow remote configuration of the storage size.
hrStorageUsed	1.3.6.1.2.1.25.2.3.1.6	Integer Type		The amount of the storage represented by this entry that is allocated, in units of hrStorageAllocationUnits.
hrStorageAllocationFailures	1.3.6.1.2.1.25.2.3.1.7	Counter		The number of requests for storage represented by this entry that could not be honored due to not enough storage. It should be noted that this counter is not reset when the storage is reconfigured.
hrDeviceTable	1.3.6.1.2.1.25.3.2			The (conceptual) table of devices contained by the host.
hrDeviceEntry	1.3.6.1.2.1.25.3.2.1			A (conceptual) entry for one device contained by the host. As an example, an instance of the hrDeviceType object might be named hrDevice1.
hrDeviceIndex	1.3.6.1.2.1.25.3.2.1.1	Integer Type		A unique value for each device contained by the host. The value for each device must remain constant at least from one re-initialization of the host to the next.
hrDeviceType	1.3.6.1.2.1.25.3.2.1.2	OID		An indication of the type of device. If this value is 'hrDeviceProcessor' then an entry exists in the hrProcessorTable which describes the processor.
hrDeviceDescr	1.3.6.1.2.1.25.3.2.1.3	String Type		A textual description of this device, including the device's manufacturer and revision, and optionally, its serial number.
hrDeviceID	1.3.6.1.2.1.25.3.2.1.4	OID		The product ID for this device.
hrDeviceStatus	1.3.6.1.2.1.25.3.2.1.5	Integer Type		The current operational state of the device described by this row of the table. A value unknown(1) indicates that the current state of the device is unknown.
hrDeviceErrors	1.3.6.1.2.1.25.3.2.1.6	Counter		The number of errors detected on this device. It should be noted that as this object has a SYNTAX of Counter32, that it does not have a default value.
hrProcessorTable	1.3.6.1.2.1.25.3.3			The (conceptual) table of processors contained by the host. Note that this table is potentially sparse: a (conceptual) entry exists only if the host has a processor of that type.
hrProcessorEntry	1.3.6.1.2.1.25.3.3.1			A (conceptual) entry for one processor contained by the host. The hrDeviceIndex in the index represents the entry in the hrDeviceTable that describes the device.
hrProcessorFirmwareID	1.3.6.1.2.1.25.3.3.1.1	OID		The product ID of the firmware associated with the processor.

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.

Step 2 Log in to the station, as required.

When you are logged into the station, the MIB file appears in the browser.

Step 3 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.

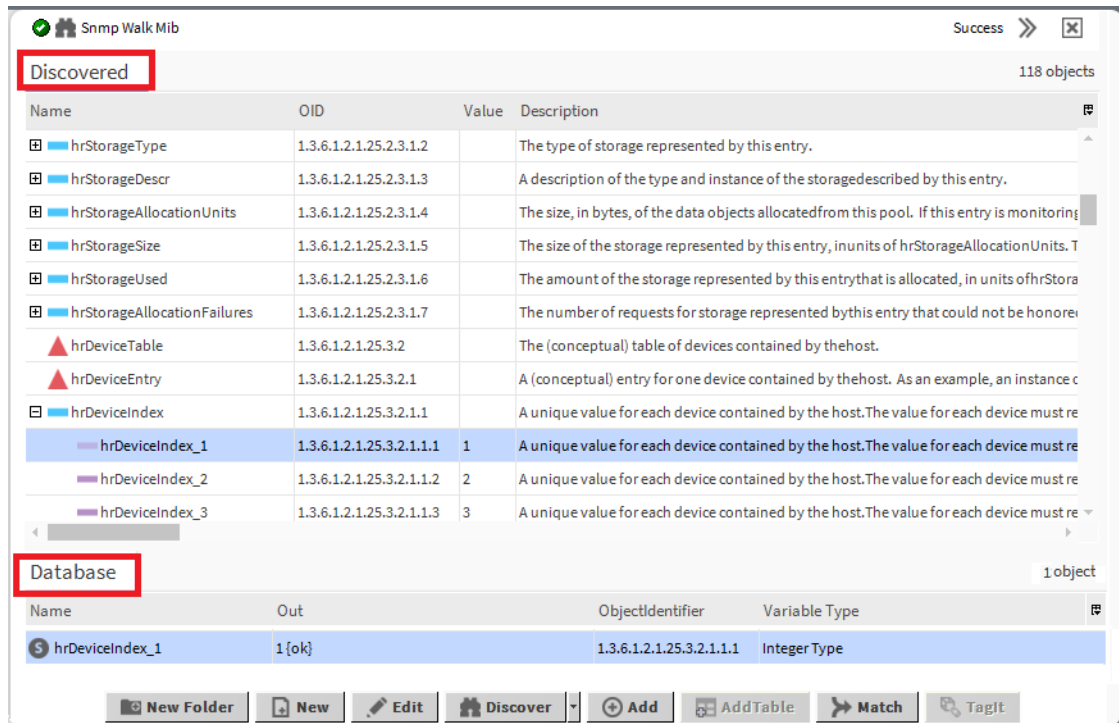
The manager view sections provide the descriptions of the different manager views that are available on the Agent (**Local Device**) and Manager (client) devices.

About the Snmp Point Manager view

The **Snmp Point Manager** view is the default view on **SnmpDevice Points** 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 **SnmpDevice** (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 *Niagara Drivers Guide*.

The **Snmp Point Manager** view splits into two panes when you select a **Discover** button and walk a MIB.



Discovered

This pane displays MIB entries from the last compiled MIB file.

Database

This pane displays the points that are added to your station database.

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, page 45](#)
- [About locally generated MIBs, page 39.](#)

About the Snmp Point Manager Discovered pane

The **Discovered** 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 are as given in the figure.

Discovered 114 objects									
Name	OID	Syntax	Type	Value	DisplayHint	Access	Status	ElementType	Description
hrNetworkIndex	1.3.6.1.2.1.25.3.4.1.1	InterfaceIndexOrZero	Integer Type		d	Read Only	current	Sequence Element	The value of ifIndex which corr
hrSWInstalledDate	1.3.6.1.2.1.25.6.3.1.5	DateAndTime	String Type		2d-1d-1d,1d:1d	Read Only	current	Sequence Element	The last-modification date of tl
hrFSLastFullBackupDate	1.3.6.1.2.1.25.3.8.1.8	DateAndTime	String Type		2d-1d-1d,1d:1d	Read Write	current	Sequence Element	The last date at which this com
hrFSLastPartialBackupDate	1.3.6.1.2.1.25.3.8.1.9	DateAndTime	String Type		2d-1d-1d,1d:1d	Read Write	current	Sequence Element	The last date at which a portior
hrSystemDate	1.3.6.1.2.1.25.1.2.0	DateAndTime	String Type		2d-1d-1d,1d:1d	Read Write	current	Nonsequence	The host's notion of the local d
hrStorageDescr	1.3.6.1.2.1.25.2.3.1.3	DisplayString	String Type		255a	Read Only	current	Sequence Element	A description of the type and ir
hrDeviceDescr	1.3.6.1.2.1.25.3.2.1.3	DisplayString	String Type		255a	Read Only	current	Sequence Element	A textual description of this dei
hrPartitionEntry	1.3.6.1.2.1.25.3.7.1	HrPartitionEntry				Not Accessible	current	Sequence Entry	A (conceptual) entry for one pe
hrPartitionTable	1.3.6.1.2.1.25.3.7	SEQUENCE				Not Accessible	current	Sequence	The (conceptual) table of partil
hrPartitionLabel	1.3.6.1.2.1.25.3.7.1.2	InternationalDisplayString	String Type			Read Only	current	Sequence Element	A textual description of this pai
hrPartitionIndex	1.3.6.1.2.1.25.3.7.1.1	Integer32	Integer Type			Read Only	current	Sequence Element	A unique value for each partit
hrStorageEntry	1.3.6.1.2.1.25.2.3.1	HrStorageEntry				Not Accessible	current	Sequence Entry	A (conceptual) entry for one lo

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, Display-String, 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.

DisplayHint

This column presents the value of an Instance of an Snmp Object with the Syntax. It is more useful for the Octet String type of Data. DisplayHint "d" is used for Decimal data. For DataAndTime object, DisplayHint is defined as "2d-1d-1d,1d:1d:1d,1a1d:1d", It parses the Byte data into "YYYY-M-D,TIME,ZONE".

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 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 **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 are as given in the figure.

Database 13 objects

Path	Name	Type	Out	ObjectIdentifier	Variable Type	Enabled	Device Facets	Facets	Conversion	Read Value	Write Value	Tuning Policy Name	Fault Cause
	/Drivers/S hrSystemUptir	String Point	28 hours	1.3.6.1.2.1.25.1.1.0	Timeticks	true	units=null,precis	units=null, Default		28 hours 49 min - {ok}	{null} @ def	defaultPolicy	
	/Drivers/S hrDiskStorage/	Enum Point	readOnly	1.3.6.1.2.1.25.3.6.1.1	Integer Type	true	range=[readWrite	range=[rea Default		readOnly {ok}	0 {ok}	defaultPolicy	
	/Drivers/S hrSystemDate	String Writabl	2016-3-25	1.3.6.1.2.1.25.1.2.0	String Type	true			Default	2016-3-29,15:15 - {null} @ def	{null} @ def	defaultPolicy	
	/Drivers/S hrSystemInitia	Numeric Writa	0.0 {ok}	1.3.6.1.2.1.25.1.3.0	Integer Type	true	units=null,precis	units=null, Default		0.0 {ok}	- {null} @ def	defaultPolicy	
	/Drivers/S hrSystemInitia	String Writabl	{ok} @ d	1.3.6.1.2.1.25.1.4.0	String Type	true			Default	{ok}	- {null} @ def	defaultPolicy	
	/Drivers/S hrSystemNum	Numeric Poin	4.0 {ok}	1.3.6.1.2.1.25.1.5.0	Gauge	true	units=null,precis	units=null, Default		4.0 {ok}	0.0 {ok}	defaultPolicy	
	/Drivers/S hrSystemProce	Numeric Poin	88.0 {ok}	1.3.6.1.2.1.25.1.6.0	Gauge	true	units=null,precis	units=null, Default		88.0 {ok}	0.0 {ok}	defaultPolicy	
	/Drivers/S hrSystemMaxP	Numeric Poin	0.0 {ok}	1.3.6.1.2.1.25.1.7.0	Integer Type	true	units=null,precis	units=null, Default		0.0 {ok}	0.0 {ok}	defaultPolicy	
	/Drivers/S hrMemorySize	Numeric Poin	1.0 {ok}	1.3.6.1.2.1.25.3.2.1.1	Integer Type	true	units=null,precis	units=null, Default		1.0 {ok}	0.0 {ok}	defaultPolicy	
	/Drivers/S hrDeviceIndex	String Point	1 {ok}	1.3.6.1.2.1.25.3.2.1.1	Integer Type	true			Default	1 {ok}	- {ok}	defaultPolicy	
	/Drivers/S BooleanWritab	Boolean Writa	false {fault}		Integer Type	true	trueText=true,false	trueText=false	Default	false {ok}	- {null} @ def	defaultPolicy	Read fault: illegal Object
	/Drivers/S NumericPoint	Numeric Poin	0.0 {fault}	1.3.6.1.2	Integer Type	true	units=null,precis	units=null, Default		0.0 {ok}	0.0 {ok}	defaultPolicy	Read fault: no such object
	/Drivers/S hrStorageTable	Snmp Table	Snmp Table	1.3.6.1.2.1.25.2.3				1.3.6.1.2.1.1					

New Folder

New

Edit

Discover

Add

AddTable

Match

TagIt

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

ObjectIdentifier

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 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 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.

Fault Cause

This column displays a description if the point goes into the fault.

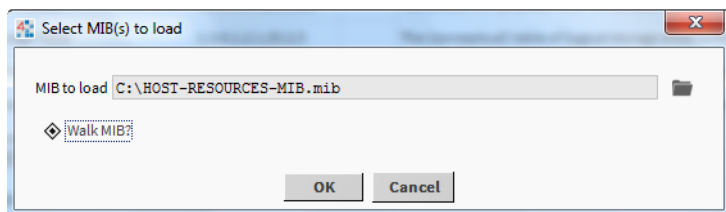
About Snmp point discovery

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.

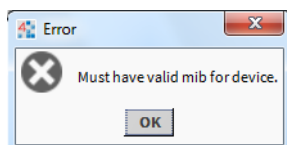
When you click the **Discover** button, the **Snmp Point Manager** opens the **Select MIB(s) to load** dialog box.



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 below.



If no errors are encountered after issuing the Load MIB command, the MIB entries contained in the loaded MIB file appear in the table.

Loading the MIB

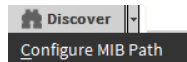
“Walking the MIB is where communication takes place between an agent (client) application and a manager application to discover the point values for those loaded point types that have values available.

Configuring dependent MIBs

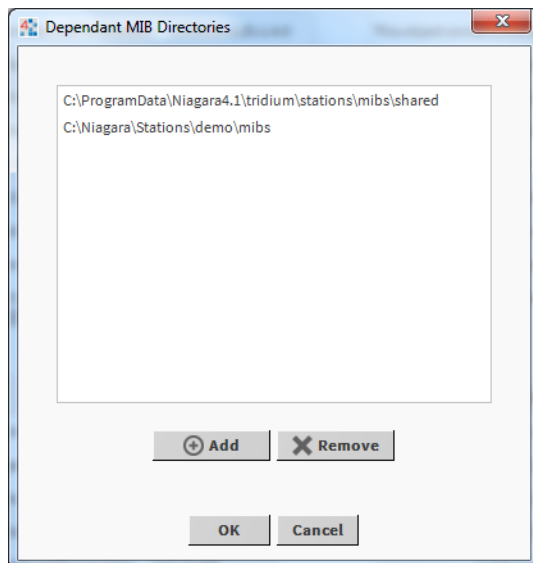
Dependent MIBs may be required by the primary MIB. Any MIBs that are not already available in the snmp module may be placed in one or more accessible folders.

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 they are loaded any time that a Load MIB type command is issued (for example, Load MIB and Walk MIB).



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:

- Click the **Add** button to navigate to a directory to add to the 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, Type, 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.

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

SnmpNetwork : SnmpDevice : Points : hrStorageTable Snmp Table Manager

Database 4 objects

hrStorageIndex	hrStorageType	hrStorageDescr	hrStorageAl
1.00 [ok]	1.3.6.1.2.1.25.2.1.4 {o	C:\Label:OSDisk Serial Number be12b2c7 {o	4096.00 [ok]
2.00 [ok]	1.3.6.1.2.1.25.2.1.7 {o	D:\ [ok]	0.00 [ok]
3.00 [ok]	1.3.6.1.2.1.25.2.1.3 {o	Virtual Memory [ok]	65536.00 [ol
4.00 [ok]	1.3.6.1.2.1.25.2.1.2 {o	Physical Memory [ok]	65536.00 [ol

WalkMib TagIt

About the Snmp Trap Manager view

The **Snmp Trap Manager** view is the default view on the **SnmpAlarmDeviceExt**. 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 **Discovered** pane to the **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.

Discovered 1 objects

Name	OID	Value	Description
printerV2Alert	1.3.6.1.2.1.43.18.2.0.1		GENERIC=6; SPECIFIC=1; OBJECTS=prtAlertIndex,prtAlertSeverityLevel,prtAlertGroup,prtAlertGrou

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	
Egg Neighbor Loss	eggNeighborLoss	1.3.6.1.6.3.1.1.5.6	5	
printerV2Alert	printerV2Alert	1.3.6.1.2.1.43.18.2.0.1	6	prtAlertIndex,prtAlertSeverityLevel,prtAlertGroup,prtAlert

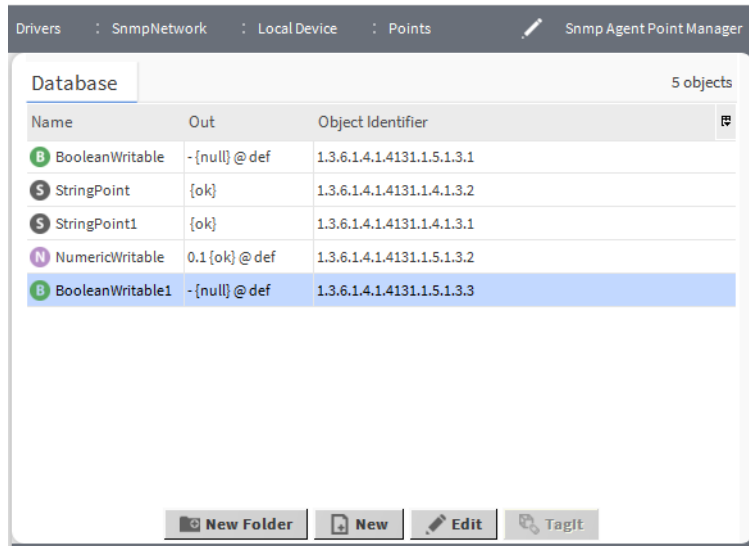
New Edit Discover Cancel Add Match TagIt

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 **Local 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.

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 **nSnmp Palette** or by using the **New** button at the bottom of the

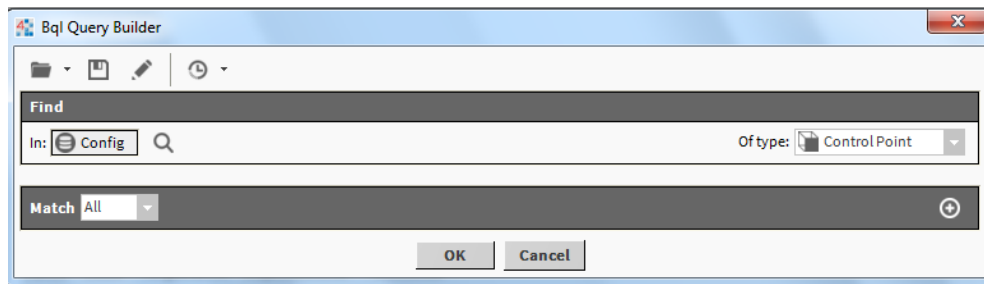
view. Use the **New Folder** button to organize and add hierarchy to your point collection. Use the **Edit** button to change a selected point's property values.



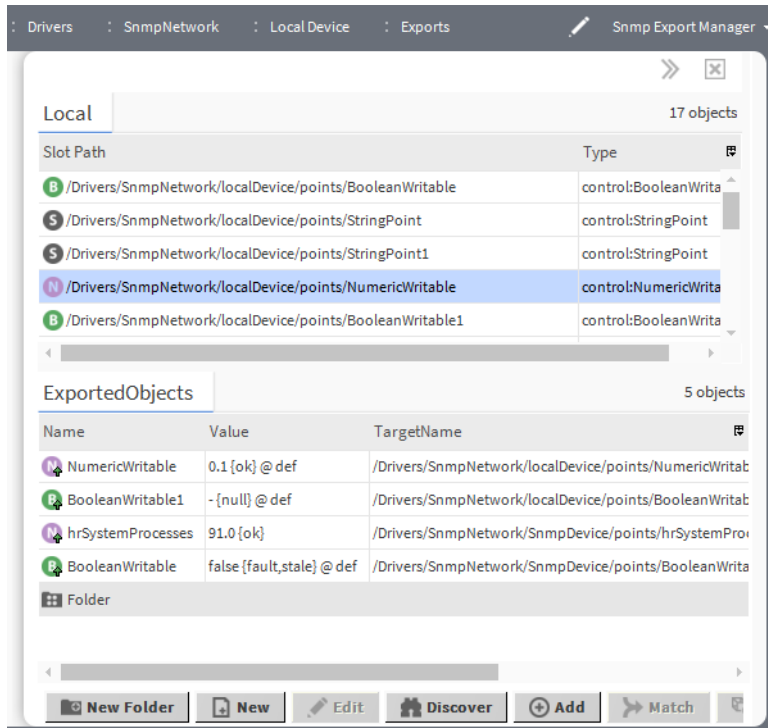
About the Snmp Export Manager (SnmpExportTable)

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

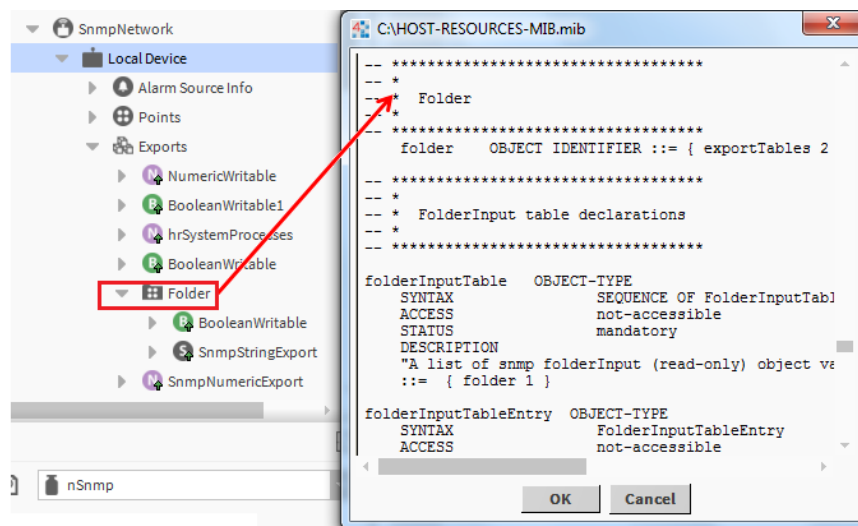
The **Snmp Export Manager** view is similar to the **Snmp Point Manager** view, however, in this view, the **Discover** button launches the **Bql Query Builder** dialog box, instead initiating a MIB loading and walking process. The **Bql Query Builder** dialog box lets you “discover” components to add to the **Localpane**, 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** pane using the **Add** or **Match** button.



After adding points to the **ExportedObjects** pane, the points appear under the **Exports** 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.

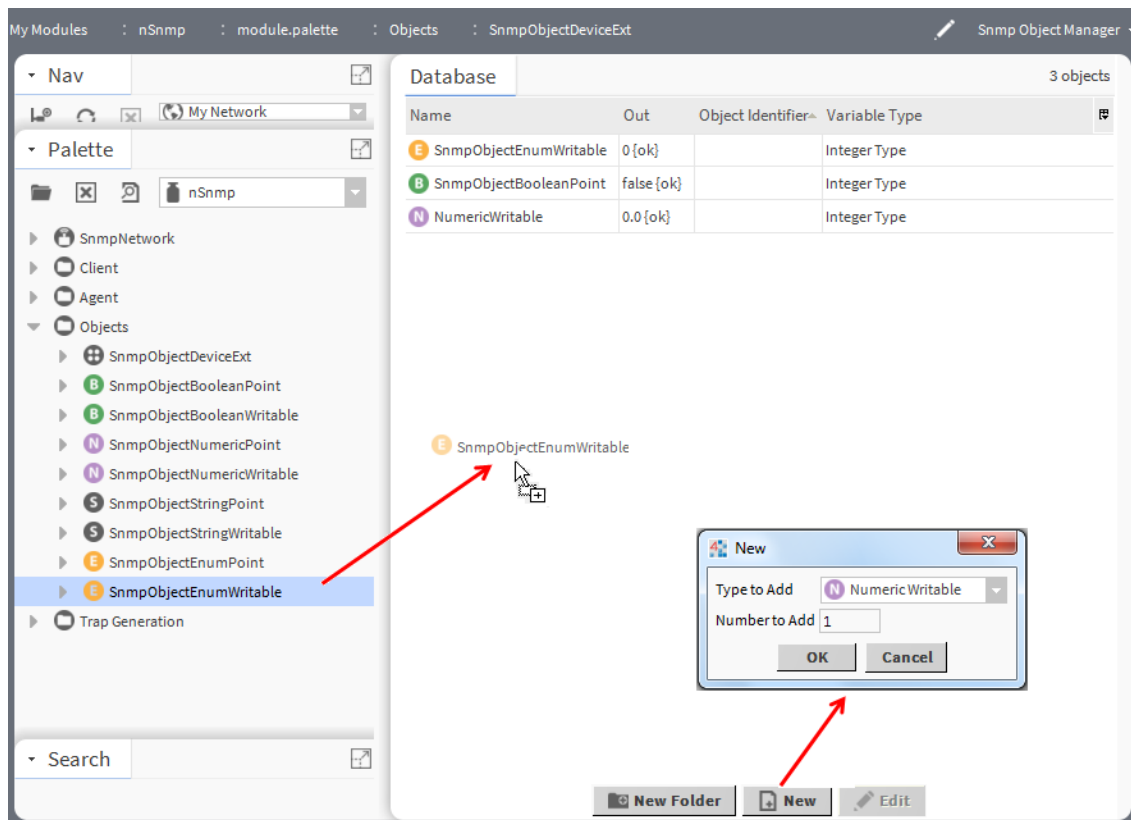


About the Snmp Object Manager view

The **Snmp Object Manager** view is the default view on the **SnmpObjectDeviceExt**. 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 **SnmpObjectDeviceExt**.

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 **nSnmp Palette**) onto the view to add them.



After adding points to the **Database** pane, the points appear under the **SnmpObjectDeviceExt** 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**.

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.

My Host: IE67DTP42WQ1.global.ds.honeywell.com (SnmpNetwork) : My Spy : logSetup

Local Workbench | logSetup

Log Configuration											
Log	Level	OFF	SEVERE	WARNING	INFO	CONFIG	FINE	FINER	FINEST	ALL	DEFAULT
DEFAULT	INFO	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Not Applicable
alarm	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
alarm.database	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
backup	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
bacnet	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
bacnet.client	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
bacnet.cov	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
bacnet.link.ethernet	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
bacnet.link.ip	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
bacnet.network	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
bacnet.point	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
bacnet.server	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
bacnet.transport	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
bacnet.virtual	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
baja	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
baja.PropertyCursor	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
lbaiaui	DEFAULT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Chapter 3 Snmp Plugins

Topics covered in this chapter

- ◆ Snmp Agent Point Manager
- ◆ Snmp Device Manager
- ◆ Snmp Export Manager
- ◆ Snmp Object Manager
- ◆ Snmp Point Manager
- ◆ Snmp Table Manager
- ◆ Snmp Trap Manager

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** bar or pressing F1 while the Plugin is selected.

The following topics describe Snmp plugins:

Snmp Agent Point Manager

Use the **Snmp Agent Point Manager** to create, edit, access, and delete SnmpAgent proxy points under a SnmpAgent.

The **Snmp Agent Point Manager** is the default view for the SnmpAgentPointDeviceExt (**Points** container) under an SnmpAgent. The **Snmp Agent Point Manager** 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 Device Manager

Use the **Snmp Device Manager** to create, edit, and view both **SnmpDevice** and SnmpAgent under a SnmpNetwork.

The **Snmp Device Manager** is the default view on the SnmpNetwork. To view, right-click SnmpNetwork and select **Views→ Snmp Device Manager**.

Snmp Export Manager

The **Snmp Export Manager** is the default view of the Snmp Export Table component.

The view has an upper (**Discovered**) 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**.

Snmp Object Manager

The **Snmp Object Manager** is the default view of the **SnmpObjectDeviceExt**.

This view provides a single database pane that allows you to add Snmp Object 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 **nSnmp Palette**.

Snmp Point Manager

Use the **Snmp Point Manager** to create, edit, access, and delete Snmp proxy points under a **SnmpDevice**.

The **Snmp Point Manager** is the default view for the SnmpPointDeviceExt (**Points** container) under an **SnmpDevice**. The **Snmp Point Manager** 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 Table Manager

The Snmp Table Manager 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 Trap Manager

Use the **Snmp Trap Manager** to discover, compile, display, and store traps from MIB files for an **SnmpDevice**.

The **Snmp Trap Manager** 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 4 Snmp Components

Topics covered in this chapter

- ◆ MIB List Table
- ◆ Snmp Agent
- ◆ Snmp Agent Boolean Proxy Ext
- ◆ Snmp Agent Numeric Proxy Ext
- ◆ Snmp Agent Point Device Ext
- ◆ Snmp Agent Point Folder
- ◆ Snmp Agent String Proxy Ext
- ◆ Snmp Alarm Device Ext
- ◆ Snmp Boolean Export
- ◆ Snmp Boolean Proxy Ext
- ◆ Snmp Enum Export
- ◆ Snmp Export Folder
- ◆ Snmp Export Table
- ◆ Snmp Numeric Export
- ◆ Snmp Object Device Ext
- ◆ Snmp Boolean Object Ext
- ◆ Snmp Numeric Object Ext
- ◆ Snmp String Object Ext
- ◆ Snmp Enum Object Ext
- ◆ Snmp Sequence
- ◆ Snmp String Export
- ◆ Snmp Table
- ◆ Snmp Table Row
- ◆ SnmpDevice
- ◆ Snmp Device Folder
- ◆ SnmpNetwork
- ◆ Snmp Numeric Proxy Ext
- ◆ Snmp Point Device Ext
- ◆ Snmp Point Folder
- ◆ N Poll Scheduler
- ◆ Snmp Recipient
- ◆ Snmp String Proxy Ext
- ◆ Trap Table
- ◆ snmp-TrapType

These topics provide help on common Snmp components.

Summary information is provided on components specific to the Snmp module.

MIB List Table

The **MIB List Table** captures information about MIB entries within SnmpDevices.


The **MIB List Table** is available in the nSnmp module. Bajadoc is available at [BMIBListTable.bajadoc](#).

Snmp Agent




Snmp Agent represents the local station as an Snmp agent device holding agent data that can be viewed and changed via Snmp from an outside Snmp manager. The Snmp Agent is available in the nSnmp module. Bajadoc is available at [BSnmpAgent.bajadoc](#).

Snmp Agent Boolean Proxy Ext

 Snmp Agent Boolean Proxy Ext contains information necessary to hold a boolean data value which can be set by an outside Snmp manager. Bajadoc is available at `BSnmpAgentBooleanProxyExt.bajadoc`.


Snmp Agent Numeric Proxy Ext

 Snmp Agent Numeric Proxy Ext contains information necessary to hold a float (or integer) data value which can be set by an outside Snmp manager. Bajadoc is available at `BSnmpAgentNumericProxyExt.bajadoc`.


Snmp Agent Point Device Ext

 Snmp Agent Point Device Ext is the container for Snmp agent proxy points representing Snmp agent data values. The Snmp Agent Point Device Ext is available in the `nSnmp` module. Bajadoc is available at `BSnmpAgentPointDeviceExt.bajadoc`.

Snmp Agent Point Folder

 Snmp Agent Point Folder is the Snmp implementation of a folder under a Snmp Agent's Points extension. You add such folders using the **New Folder** button in the view of the **Points** extension. Each Snmp Agent Point Folder has its own view. The Snmp Agent Point Folder is also available in the **nSnmp Palette**. Bajadoc is available at `BSnmpAgentPointFolder.bajadoc`.


Snmp Agent String Proxy Ext

 Snmp Agent String Proxy Ext contains information necessary to hold string data value which can be set by an outside Snmp manager. Bajadoc is available at `BSnmpAgentStringProxyExt.bajadoc`.


Snmp Alarm Device Ext

 Snmp Alarm Device Ext contains the standard generic Snmp trap-types. These are: Cold Start, Warm Start, Link Down, Link Up, Authentication Failure, EGP Neighbor Loss.

Snmp Boolean Export


 The Snmp Boolean Export is a component that exports a boolean value from an SnmpNetwork **Local Device** to an SnmpNetwork Manager (Client). Add Snmp export components under the Snmp Export Table or under folders that can be nested under the Snmp Export Table. Use the **Snmp Export Manager** view to discover control points in your station and to add them to the Snmp Export Table.

Snmp Boolean Proxy Ext


 Snmp Boolean Proxy Ext contains information necessary to read a boolean data value from an **SnmpDevice**. 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 Snmp Boolean Proxy Ext to describe how to read the point. Bajadoc is available at `BSnmppBooleanProxyExt.bajadoc`.


Snmp Enum Export

 The Snmp Enum Export is a component that exports an enumerated value from an SnmpNetwork **Local Device** to an SnmpNetwork Manager (Client). Add Snmp export components under the Snmp Export Table or under folders that can be nested under the Snmp Export Table. Use the **Snmp Export Manager** view to discover control points in your station and to add them to the Snmp Export Table.


Snmp Export Folder

 The Snmp Enum Export is a component that exports an enumerated value from an SnmpNetwork **Local Device** to an SnmpNetwork Manager (Client). Add Snmp export components under the Snmp Export Table or under folders that can be nested under the Snmp Export Table. Use the **Snmp Export Manager** view to discover control points in your station and to add them to the Snmp Export Table.


Snmp Export Table

 Snmp Export Table 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 Snmp Export Table.

Snmp Numeric Export

 The Snmp Numeric Export is a component that exports a numeric value from an SnmpNetwork **Local Device** to an SnmpNetwork Manager (Client). Add Snmp export components under the Snmp Export Table or under folders that can be nested under the Snmp Export Table. Use the **Snmp Export Manager** view to discover control points in your station and to add them to the Snmp Export Table.

Snmp Object Device Ext

 Snmp Object Device Ext is a container for any of the set of Snmp Object 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 Boolean Object Ext

An Snmp Boolean Object Ext is the container for SnmpObjectBooleanPoint and SnmpObjectBooleanWritable objects. The Snmp Boolean Object Ext is available in the snmp module.

Snmp Numeric Object Ext

An Snmp Numeric Object Ext is the container for SnmpObjectNumericPoint and SnmpObjectNumericWritable objects. The Snmp Numeric Object Ext is available in the snmp module.

Snmp String Object Ext

An Snmp String Object Ext is the container for SnmpObjectStringPoint and SnmpObjectStringWritable objects. The Snmp String Object Ext is available in the snmp module.

Snmp Enum Object Ext

An Snmp Enum Object Ext is the container for SnmpObjectEnumPoint and SnmpObjectEnumWritable objects. The Snmp Enum Object Ext is available in the snmp module.

Snmp Sequence



Snmp Sequence objects contain an Output Index, Output Name, and Output Value slot (properties) that correspond to the columns in each **Snmp Table Row**. Each of these has the following editable properties: OID, Element Name, Element Type, Variable Type, Writable option and Display Hint.

Snmp String Export



The Snmp String Export is a component that exports a string value from an SnmpNetwork **Local Device** to an SnmpNetwork Manager (Client). Add Snmp export components under the Snmp Export Table or under folders that can be nested under the Snmp Export Table. Use the **Snmp Export Manager** view to discover control points in your station and to add them to the Snmp Export Table.

Snmp Table



The Snmp Table 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 Snmp Table object is the **Table Manager** view, which presents the data in a table format. The row elements can be linked to other control objects. Read-only values can be linked to inputs and read-write values can be linked to inputs or outputs.

Snmp Table Row



The Snmp Table Row contains an object for each element that is represented in the MIB table for the **SnmpDevice**. The row has an index, name, type, and value.

SnmpDevice




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

Snmp Device Folder




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

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 nSnmp module. Bajadoc is available at [BSnmpNetwork.bajadoc](#).

Snmp Numeric Proxy Ext


 Snmp Numeric Proxy Ext contains information necessary to read a float (or integer) data value from a **SnmpDevice**.

Each read-only proxy point that represents a readable float (integer) Snmp data quantity will have a Snmp Numeric Proxy Ext to describe how to read the point. Bajadoc is available at [BSnmpNumericProxyExt.bajadoc](#).


Snmp Point Device Ext

 Snmp Point Device Ext is the container for Snmp proxy points representing **SnmpDevice** data values. The Snmp Point Device Ext is available in the nSnmp module. Bajadoc is available at [BSnmpPointDeviceExt.bajadoc](#).


Snmp Point Folder

 Snmp Point Folder is the Snmp implementation of a folder under a **SnmpDevice Points** extension. You add such folders using the **New Folder** button in the **Snmp Point Manager** view of the Snmp Point Device extension. Each Snmp Point Folder has its own view (**Snmp Point Manager** view). The Snmp Point Folder is also available in the **nSnmp Palette**. Bajadoc is available at [BSnmpPointFolder.bajadoc](#).


N Poll Scheduler

 N Poll Scheduler. The N Poll Scheduler is available in the nSnmp module. Bajadoc is available at [BSnmpPollScheduler.bajadoc](#).


Snmp Recipient

 Snmp Recipient recipient class is used to send alarm traps to an SnmpNetwork manager and to store alarms until acknowledged. The Snmp Recipient is available in the nSnmp module. Bajadoc is available at [BSnmpRecipient.bajadoc](#).

Snmp String Proxy Ext

 Snmp String Proxy Ext contains information necessary to read a String data value from a **SnmpDevice**. Each read-only proxy point that represents a readable string Snmp data quantity will have an Snmp String Proxy Ext to describe how to read the point. Bajadoc is available at [BSnmpStringProxyExt.bajadoc](#).

Trap Table

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

snmp-TrapType

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

