Technical Document

NRIO Driver Guide

May 16, 2017

niagara4

NRIO Driver Guide

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

Contents

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 or NiagaraAX software.

Document Content

NOTE: Content in this document covers releases NiagaraAX-3.8U2/Niagara 4.3 and later.

This document describes procedures to install and configure the Niagara Remote I/O (NRIO) driver software. It also describes the procedures used to create an NrioNetwork in a Niagara 4.x station for the purpose of integrating remote I/O data into the Niagara Framework for monitor and control operation. Also included is brief information for most of the components and plugins included in the nrio module.

The intended audience of this document is the Niagara certified systems integrator.

Document Change Log

This topic provides a record of document changes.

Updates (changes/additions) to the NRIO Driver Guide document are listed below.

- Updated May 16, 2017

  Content updated to include changes resulting from development of remote I/O hardware modules, (IO–R–16 and IO–R–34) in Niagara 4.3 for use with Niagara JACE-8000 embedded controllers.

  – Throughout the document, removed outdated verbiage referring to "...remote I/O devices available at some later time," and similar phrases.

  – Many changes to topics in the Nrio Driver Installation chapter, added references to remote I/O modules for the JACE-8000, also rearranged topics and combined a few.

  – In the topic, "About the NrioNework," edited the 1st note to include the added support for remote I/O modules for the JACE-8000

  – In the topic, "About the NrioModule," added details about the support for new I/O hardware modules, and added details on driver Default Output Value changes.

  – In the Component Guides chapter, there is an added topic on the nrio-Nrio34Module component.

- Published: May 5, 2009

  Initial document.

Related documentation

This topic lists documents that are related to this guide.

- Niagara Drivers Guide
- Niagara 4 Platform Guide

Chapter 1 Nrio Driver installation

Topics covered in this chapter

♦ Configure the Nrio network
♦ Create Nrio proxy points

To use the nrio driver, you must have a target JACE host that provides either onboard I/O hardware points using Nrio, such as an M2M JACE, and/or that supports external I/O module(s) connected to onboard RS-485, such as a JACE-7 series, or a JACE-8000 (in Niagara 4.3 and later).

In addition, the `nrio` feature must be in the JACE's license.

Apart from installing the Niagara version distribution in the controller, make sure to install the **nrio** and **kitIo** modules, plus any required dependency files.

NOTE: The **kitIo** module provides a Generic Tabular conversion selection for non-linear sensor support, plus has several types of thermistor curve files in its xml folder.

Upgrade any modules shown as "out of date". For instructions about updating your modules, see "Software Manager" in the *Niagara 4 Platform Guide*.

The remote JACE is now ready for Nrio software configuration in its running station, as described in the next section.

NOTE: Basic procedures for using the Nrio driver, including online discovery of I/O points (and their addition to your station), is in the next section.

Configure the Nrio network

Two types of Nrio networks are available. The **`M2mIoNetwork`** supports the onboard I/O of an M2M JACE (JACE-202 Express). And the **`NrioNetwork`** supports external I/O module(s) connected to onboard RS-485 of either a JACE-7 series or JACE-8000 (in Niagara 4.3 and later).

A separate NrioNetwork is required for each RS-485 trunk with remote I/O modules.

Each Nrio network must have the following:

- A unique, specific **`Port Name`** property `COM n` value.
- A unique **`Trunk`** property value, starting at `1`, then `2`, and so on.

For example: In an M2M JACE where both the onboard I/O is used and one or more remote I/O modules are wired to the onboard RS-485/power port, two different Nrio network components are needed:

- M2mIoNetwork (for onboard I/O): **`Port Name`** to `COM3`, **`Trunk`** to `1` (both are fixed, read-only values)
- NrioNetwork (onboard RS-485 connected to remote I/O modules): **`Port Name`** to `COM2`, **`Trunk`** to `2`.

Adding a M2mIoNetwork in the station

Use the following procedure to add an M2mIoNetwork under the station's **Drivers** container.

Step 1 In Workbench, open the M2M JACE station.

Step 2 Expand the station's Config space to see its **Drivers** folder, and double-click it for the **Driver Manager** view. If an M2mIoNetwork is already there, verify it is enabled (status "ok"), and go to Step 5.

Step 3 Open the **nrio** palette from the palette side bar.

Step 4 From the **nrio** palette, drag (or copy and paste) an M2mIoNetwork into the station's **Drivers** folder. In the popup **Name** dialog box, you can rename the network—or, simply use the default name.

An M2mIoNetwork named M2mIoNetwork (or whatever you named it), is under your **Drivers** folder. As copied from the palette, the M2mIoNetwork includes a child **Nrio16Module** named **LocalIo16**.

Step 5 Double-click the M2mIoNetwork to open its **Nrio Device Manager** view and begin discovery.

NOTE: If the station is running, within seconds the previous copy triggers a discover job, and the **LocalIo16** Nrio16Module dynamically acquires its address and UID. At this point, its status should be "ok". If the **LocalIo16 Nrio16Module** remains in fault, go to its property sheet and look at the `Fault Cause` property.

Step 6 Click to select all discovered **Nrio16Modules**, then click the **Add** button .

The **Add** dialog box appears, in which you can enter a Description for each one, or edit that later.

Step 7 Click **OK** to add the **Nrio16Module**(s) to your station.

Once the discovered modules are added, you can proceed to create Nrio proxy points.

Adding an NrioNetwork in the station

Use the following procedure to add an NrioNetwork under the station's **Drivers** container.

Step 1 Double-click the station's **Drivers** container, to bring up the **Driver Manager**.

Step 2 Click the **New** button to bring up the **New DeviceNetwork** dialog box.

Step 3 Select NrioNetwork, number to add: 1, and click **OK**.

This brings up a dialog to name the network. Typically you enter a descriptive name, such as Remote_IO or something similar.

Step 4 Click **OK** to add the NrioNetwork to the station.

A NrioNetwork named NrioNetwork (or whatever you named it), is under your **Drivers** folder.

Step 5 Open the NrioNetwork's property sheet, and edit `Port Name` and `Trunk` values as needed.

For example, for the onboard RS-485 port of an M2M JACE, set `Port Name` to `COM2` and `Trunk` to `2`.

If necessary, add, configure, and save any additional NrioNetwork needed in a similar fashion.

Step 6 Click on the **Save** button.

Step 7 After the JACE reboots and the station finishes starting, reopen the station and expand the added NrioNetwork. You are now ready to Discover and add NrioModules.

Discovering and adding NrioModules

In Nrio architecture, NrioModules act as device-level components. An **Nrio16Module** represents an I/O processor, servicing up to 16 I/O points, and the Nrio34Module services up to 34 I/O points.

Depending on the JACE platform and installed devices, the number of **NrioModule**s to be discovered varies:

- An M2mIoNetwork for an M2M JACE's (onboard) I/O has only a single **Nrio16Module**.
- An NrioNetwork for remote I/O module (devices) connected to a specific RS-485 bus should discover the same number of NrioModules as I/O devices on that bus.

  NOTE: The limit for remote I/O module (devices) connected to a specific RS-485 bus is 16 modules on the same bus. The IO–R–16 counts as 1 module, and the IO–R–34 counts as 2 modules.

The following table lists component and applicable hardware:

NrioModule component	Used to model
LocalIo16	16 onboard I/O points of an M2M JACE
Nrio16Module	16 I/O points on a remote RS-485 I/O module (T-IO-16-485) of a JACE-7 series
Nrio16Module	16 I/O points on remote RS-485 I/O module (IO–R–16) of a JACE-8000
Nrio34Module	34 I/O points on remote RS-485 I/O module (IO–R–34) of a JACE-8000

Step 1 Double-click the NrioNetwork to display the **Nrio Device Manager** view.

An alternative method is right-clicking the NrioNetwork and select **Views→ Nrio Device Manager**).

Step 2 Click the **Discover** button to launch an Nrio Discovery job.

A progress bar appears at the top of the view, and updates as the discovery occurs.

Step 3 When the discovery job completes, discovered **NrioModules** are listed in the top pane of the view, in the Discovered table. The bottom pane, labeled **Database**, is a table of Nrio16Modules that are currently mapped into the station. Initially (unless offline programming occurred), this pane will be blank.

NOTE: Discovered NrioModules each have an available right-click Wink action, which if invoked causes the first relay output on that I/O to cycle On and Off for 10 seconds (by default). In cases where multiple NrioModules are discovered, such as with RS-485 connected remote I/O modules, this is available to identify the association of each discovered NrioModule with a specific I/O module device. As you cannot manually edit the NrioModule's address, even after adding it to the database, using Wink is particularly important if you previously added NrioModules offline (Added Offline Hardware), and are using the Match feature. After adding an NrioModule, note that the Wink action is still available on the component.

Step 4 Click to select all discovered **Nrio16Modules**, then click the **Add** button .

The **Add** dialog box appears, in which you can enter a Description for each one, or you can enter descriptions at a later time.

Step 5 Click **OK** to add the **Nrio16Module**(s) to your station.

NOTE: If online with the station, but I/O devices are not yet available, you can use the Add Offline Hardware function of the **Nrio Device Manager** to add Nrio16Modules. You can then do a point discover under each **Nrio16Module**, and add points offline.

Once the discovered modules are added, you can proceed to create Nrio proxy points.

Create Nrio proxy points

As with device objects in other drivers, each **NrioModule** has a points extension that serves as the container for proxy points. The default view for any points extension is the **Point Manager** (and in this case, the **Nrio Point Manager**). You use it to create Nrio proxy points under any **NrioModule**.

Use the following procedures to add Nrio points.

NOTE: If your Nrio network has multiple **NrioModule**s, repeat all procedures for each **NrioModule**, until you have all I/O points proxied in the station.

Discovering I/O points

Perform this task to discover I/O points.

Step 1 In the **Nrio Device Manager**, in the **Exts** column, double-click on the **Points** in the row representing the **NrioModule** you wish to explore.

This brings up its **Nrio Point Manager**.

Step 2 Click the **Discover** button to learn what I/O points are available on the **NrioModule**.

When the discovery job completes, discovered I/O points are listed in the top pane of the view, in the Discovered **Hardware Points** table. Each I/O point occupies one row.

Adding Nrio point folders (optional)

This optional procedure adds `Nrio Point Folders` for the purpose of containing and organizing Nrio proxy points. Using point folders to organize proxy points can facilitate working with points. Note that point folders can be added at any time, and proxy points moved into them, if needed. However, adding point folders before adding points can be a time saver.

Step 1 Click the **New Folder** button in the **Nrio Point Manager** to add an `Nrio Point Folder`.

Step 2 In the popup **Name** dialog, enter a name for the folder and click **OK**.

NOTE: In general, short names are recommended to keep ord lengths manageable.

The `Nrio Point Folder` is added under the **Points** container, at the current hierarchy level.

Step 3 As needed, repeat steps 1 and 2 to add more Nrio point folders, either at the current level under the **Points** container, or after double-clicking an `Nrio Point Folder` to move down one level in folder structure.

NOTE: Changes made in the **Nrio Point Manager** improve point folder usage when adding discovered Nrio proxy points. For instance, you can select (highlight) a point folder in the lower **Database** pane, then add one or more discovered points (from the top pane) directly to that folder.

Adding discovered I/O points as Nrio proxy points

NOTE: Only one Nrio proxy point can be added for each discovered **Hardware Point**. This prevents input configuration errors and output write contentions to the I/O hardware points.

Step 1 Select the I/O point or points in the discovered **Hardware Points** pane of the **Nrio Point Manager**.

Step 2 (Optional) Select a target `Nrio point` folder in the **Database** pane (click to highlight folder).

Step 3 Map selected points in the station by either of the follow methods:

- Dragging from the **Discovered** pane to **Database** pane which invokes the **Add** window.
- Double-clicking an item in the **Discovered** pane also invokes the **Add** window.

Step 4 In the **Add** window, edit the following fields for each I/O point, as needed.

- **`Name`** is the Nrio proxy point name—by default this appears as AbbrevIoTypeTerminalNumber, for example, `ui4`, `ao2`, and so on. Typically, you change this to describe the I/O purpose.
- **`Type`** is the Nrio type, which is selectable for any universal input, but fixed for any output.

  NOTE: Unlike other entries in the **Add** dialog, you cannot edit the point's Type later.
- **`Address`** is the learned I/O point's hardware address.
- **`Poll Frequency`** specifies the point's poll frequency group (default is Normal).
- **`Conversion`** specifies the conversion type used between the Nrio proxy extension's Device Facets and the parent point's facets.
- **`Facets`** are the Nrio proxy point's facets, for how the value should be displayed in Niagara.

Step 5 Click **OK** once the proxy point(s) are configured properly for your usage.

The proxy points ares added to the station, and appear listed in the **Database** pane.

NOTE: In cases where you notice a fault status for input points you have edited with non-default facets, this is likely related to a mismatch of the Units in the point's automatically-appended **Linear Calibration Ext**. To clear this fault status, expand the point's Linear Calibration Ext and edit Units to match the units in the point's facets.

Chapter 2 Niagara Nrio Concepts

Topics covered in this chapter

- ♦ About Nrio Architecture
- ♦ About the Nrio network
- ♦ Nrio Device Manager
- ♦ About the NrioModules
- ♦ Nrio Point Manager
- ♦ About Nrio proxy points

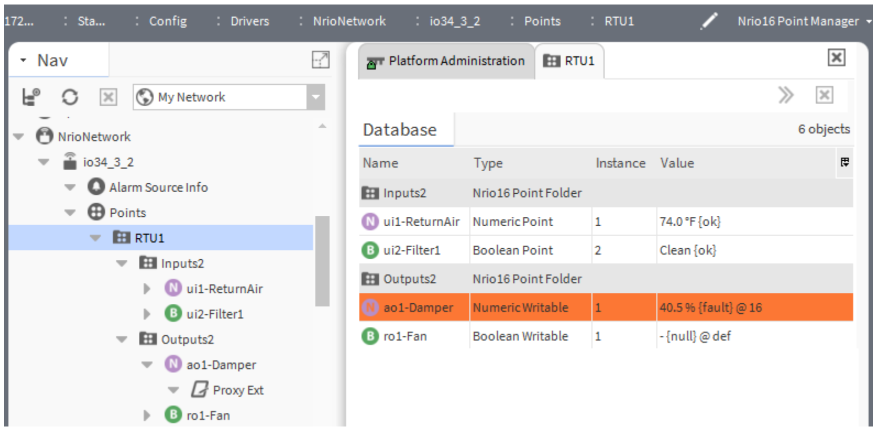
This section provides conceptual details on the Nrio driver (Remote input/output) and its components, including views. Nrio components are the station interface to certain JACE controllers with onboard I/O points, such as the M2M JACE (JACE-202 Express), and/or remote I/O expansion modules accessed over RS-485.

About Nrio Architecture

Essentially, Nrio uses the standard NiagaraNetwork architecture. For example, real-time data is modeled using Nrio proxy points, which reside under an **NrioModule** device, which in turn resides under an Nrio network container (M2mIoNetwork or NrioNetwork) in the station's DriverContainer (**Drivers**).

Hierarchically, the component architecture is: network, module, points extension, points.

Figure 1 Nrio driver architecture



Name	Type	Instance	Value
Inputs2	Nrio16 Point Folder		
ui1-ReturnAir	Numeric Point	1	74.0 °F {ok}
ui2-Filter1	Boolean Point	2	Clean {ok}
Outputs2	Nrio16 Point Folder		
ao1-Damper	Numeric Writable	1	40.5 % {fault} @ 16
ro1-Fan	Boolean Writable	1	- {null} @ def

Unlike most other drivers, points is the only extension under an **NrioModule** device, and the sole purpose of the Nrio driver—to configure (and proxy) the actual Nrio I/O hardware points.

NOTE: You use Manager views of Nrio container components to add all Nrio components to your station, including Nrio proxy points. In these views, the Nrio driver provides online discovery of available hardware (Learn Mode), which greatly simplifies engineering.

About the Nrio network

An Nrio network (NrioNetwork or M2mIoNetwork) is a top-level container component for one access path of Nrio in a station.

NOTE: In the initial nrio release, only a specialized type of Nrio network component was used: the M2mIo-Network, which supports the integral onboard I/O on an M2M JACE (JACE-202 Express). Later, remote I/O modules (T-IO-16-485) became available. And in Niagara 4.3 and later, there is added support for IO–R–16/ IO–R–34 remote I/O modules connected to a JACE-8000. These multiple Nrio access paths require multiple Nrio networks in the station (using NrioNetwork components). Essentially, a separate path (network) is required for each of the following: any JACE onboard I/O; remote I/O modules wired to the JACE's onboard RS-485/power connector; remote I/O modules wired to any RS-485 option card port; or remote I/O option modules connected to the JACE's onboard RS-485 port.

Any Nrio network should reside in the station's **Drivers** Container. The simplest way to add an NrioNetwork is from the **Driver Manager** view, using the New command. Or, you can simply copy an M2mIoNetwork or NrioNetwork from the **nrio** palette into **Drivers**. This is the recommended method for an M2M JACE station.

NOTE: The JACE platform must have the **nrio** module installed. Otherwise, an error occurs explaining that the nrio module is missing. If this occurs, install the **nrio** module in that JACE and repeat the operation.

An Nrio network component has the typical collection of slots and properties as most other network components.

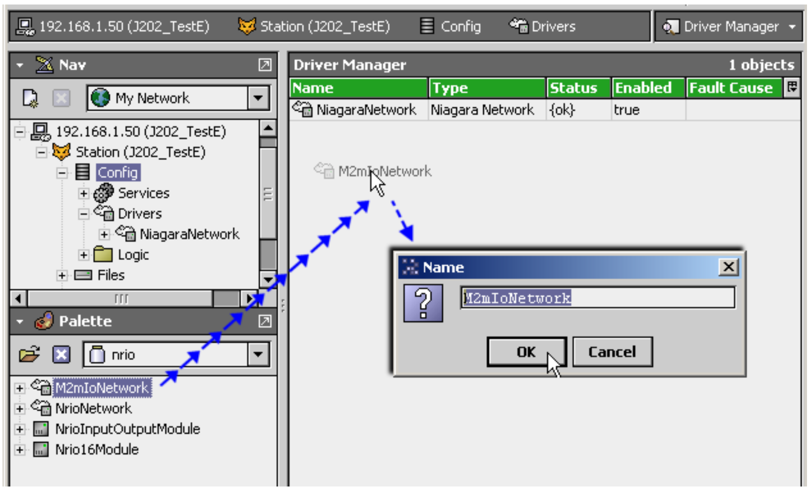
About the M2mIoNetwork

The M2mIoNetwork is a specialized version of the NrioNetwork, pre-configured to communicate to the onboard I/O processor of an M2M JACE (JACE-202 Express). Currently, it is valid only for this platform.

NOTE: Only one M2mIoNetwork is supported in a station. In the initial nrio release, this is the only IO-related network needed by an M2M JACE station. At some future time, when remote I/O modules (10-16-485) are available, one or more NrioNetworks can be separately added to support these RS-485 connected devices.

The M2mIoNetwork should reside in the station's **Drivers** Container. The recommended way is to simply copy (drag and drop) the M2mIoNetwork from the `nrio` palette into **Drivers**.

Figure 2 Dragging M2mIoNetwork from nrio palette into station's DriverContainer



Name	Type	Status	Enabled	Fault Cause
NiagaraNetwork	Niagara Network	{ok}	true	

M2mIoNetwork

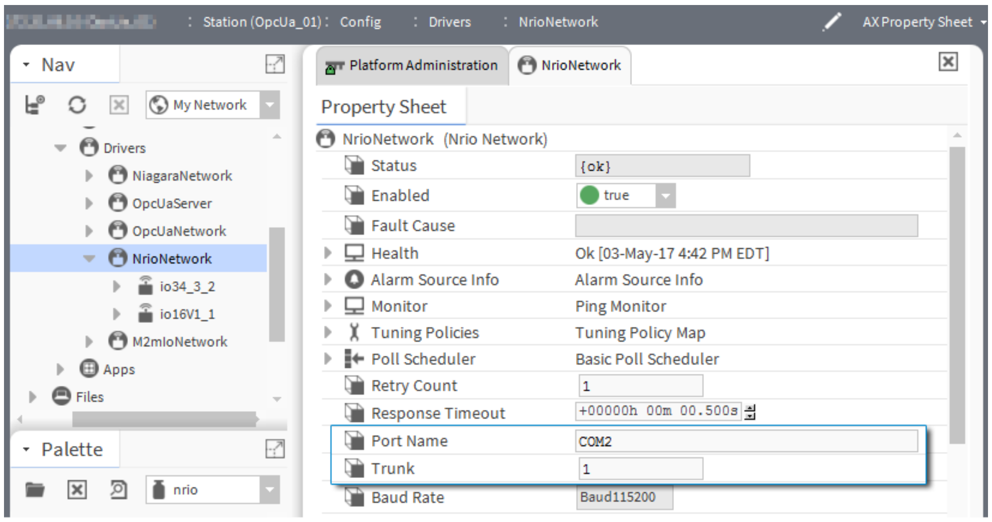
The copied M2mIoNetwork contains a child **Nrio16Module** named **LocalIo16**, which is the only device level component needed. Following the copy, a learn device job is automatically launched, and the **Nrio16Module** has its address and UID properties dynamically set.

The M2mIoNetwork and NrioNetwork components have the typical collection of slots and properties as most other network components.

Nrio network configuration properties

Two essential configuration properties for any Nrio network are the Port Name and Trunk; access these on the network's property sheet view, as shown.

Figure 3 Essential Nrio network properties Port Name, Trunk



NrioNetwork (Nrio Network)

Status	{ok}
Enabled	true
Fault Cause	
Health	Ok [03-May-17 4:42 PM EDT]
Alarm Source Info	Alarm Source Info
Monitor	Ping Monitor
Tuning Policies	Tuning Policy Map
Poll Scheduler	Basic Poll Scheduler
Retry Count	1
Response Timeout	+00000h 00m 00.500s
Port Name	COM2
Trunk	1
Baud Rate	Baud115200

- **`Port Name`**

  Each Nrio network in a station must have a unique, specific **`Port Name`** value, with the required entry depending on the JACE platform and connection, as shown in Table below.

  Table 1 Nrio network **`Port Name`** property value by JACE platform and IO connection

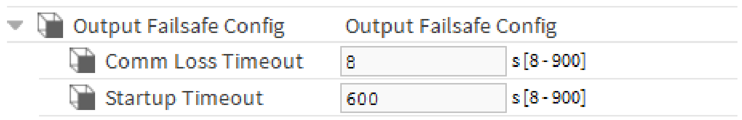
JACE platform	Onboard I/O	Onboard RS-485	RS-485 option card, Ports A & B
JACE-8000	—	COM1, COM2	—
JACE-7 series	—	COM2	COM3, COM4
M2M JACE (JACE-202 Express)	COM3	COM2	COM7, COM8

- **`Trunk`**

  Every Nrio network in a station must have a unique **`Trunk`** property value, starting at `1`, then `2`, and so on. This value corresponds to the low-level "actrldn" (access control daemon) that polls that IO.

  NOTE: These properties are writable only for an NrioNetwork component (to support remote RS-485 IO modules). Both properties are fixed and read-only for an M2mIoNetwork, as **`Port Name`** to `COM3` and **`Trunk`** to `1`.

In Niagara 4.3 and later, the Nrio network has the added Output Failsafe Config property, which you may use to define Comm Loss Timeout and Startup Timeout timer values for all child IO–16 devices.



Output Failsafe Config	Output Failsafe Config	
Comm Loss Timeout	8	s [8 - 900]
Startup Timeout	600	s [8 - 900]

NOTE: The above two timeout functions can be individually disabled in each of the child IO-16 devices. See the remote IO hardware module's **`OutputDefaultValues`** component.

Property	Value	Description
`Comm Loss Timeout`	8–900 seconds, 8 (default)	Defines the timeout value used by each child IO-16 to detect and enter the Comm Loss state. Default value is 8 seconds.
`Startup Timeout`	8–900 seconds, 600 (default)	Defines the timeout value used by each child IO-16 to detect and enter the Startup No Comm state. Default value is 600 seconds.

Nrio Network status notes

Unlike most fieldbus drivers such as Bacnet and Lonworks, the status of an Nrio network can be “down,” in addition to the normal “ok” or less typical “fault” (fault might result from misconfiguration of a network component, for example a duplicated Trunk property value). A down status might occur in the case of external I/O where the RS-485 wired connection to the JACE is broken (no communications possible).

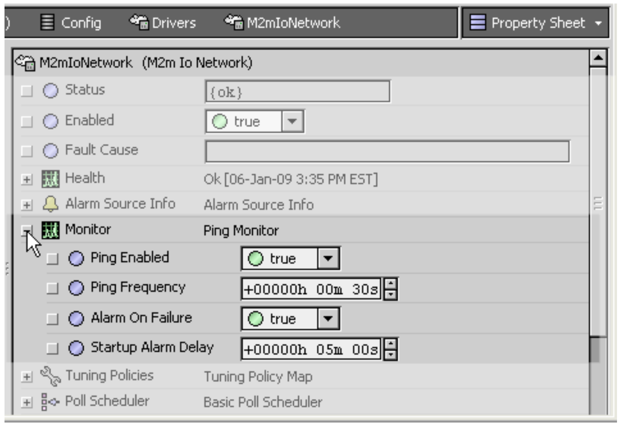
The Health slot contains historical timestamp properties that record the last network status transitions from ok to any other status. The “Fault Cause” property further explains any fault status.

NOTE: As in other driver networks, an Nrio network has an available **Alarm Source Info** container slot you can use to differentiate Nrio network alarms from other component alarms in the station.

Nrio Network monitor notes

The network’s monitor routine verifies child **NrioModule** component(s)—the pingable device equivalent in Nrio.

Figure 4 Monitor properties of an Nrio network



M2mIoNetwork (M2m Io Network)

Status	{ok}
Enabled	true
Fault Cause	
Health	Ok [06-Jan-09 3:35 PM EST]
Alarm Source Info	Alarm Source Info
Monitor	Ping Monitor
Ping Enabled	true
Ping Frequency	+00000h 00m 30s
Alarm On Failure	true
Startup Alarm Delay	+00000h 05m 00s
Tuning Policies	Tuning Policy Map
Poll Scheduler	Basic Poll Scheduler

Nrio network tuning policy notes

An Nrio network has the typical network-level Tuning Policy Map slot with a single default Tuning Policy.

However, given the number of total Nrio proxy points supported under an NrioNetwork, the importance of creating additional tuning policies is generally lower than in other networks. As most tuning policy properties mostly apply to writable proxy points, you should review and understand the default property values, most of which apply to the Nrio points for the I/O outputs (point types VoltageOutputWritable and RelayOutputWritable, for AOs and ROs, respectively).

Nrio network poll scheduler notes

An Nrio network has the typical network-level Poll Scheduler slot with standard collection of properties. The following notes apply to the Poll Scheduler of an Nrio network:

- By default, discovered/added Nrio proxy points have a **`Poll Frequency`** setting of `Normal` at a default Normal Rate of 5 seconds. If needed, you can adjust the Fast, Normal, and Slow Rates of the network's Poll Scheduler. Or, in any Nrio proxy point's ProxyExt, set its Poll Frequency from normal to either fast or slow.
- Note that with the Nrio driver, polling picks values received by the low-level access control daemon, where updates can occur two or three times a second.

Nrio network views

The M2mIoNetwork's or NrioNetwork's default view is the **Nrio Device Manager**, equivalent to the **Device Manager** in most other drivers. You use this view to discover, and in the case of an NrioNetwork, add **Nrio-Module components** to the station.

Other standard views are also available on an Nrio network. However, apart from the **Nrio Device Manager**, you typically access only its property sheet.

About the actrld (access control daemon)

For each Nrio network (M2mIoNetwork, NrioNetwork), there is an underlying "actrld" (access control daemon), a low-level process that runs on the JACE host, separate from the station. The actrld polls known Nrio devices (I/O processors) for possible value updates. Each IO device is polled at least 3 times a second, a value fixed in software.

The message response from each IO device is its IoStatus value, a concatenated collection of all IO values.

The actrld does a memory compare with the previous IoStatus message received from each device. If different, the new IoStatus is passed up to the station's **Nrio** driver. When the Nrio network polls for values, it reads from each IO device's values in its IoStatus slot.

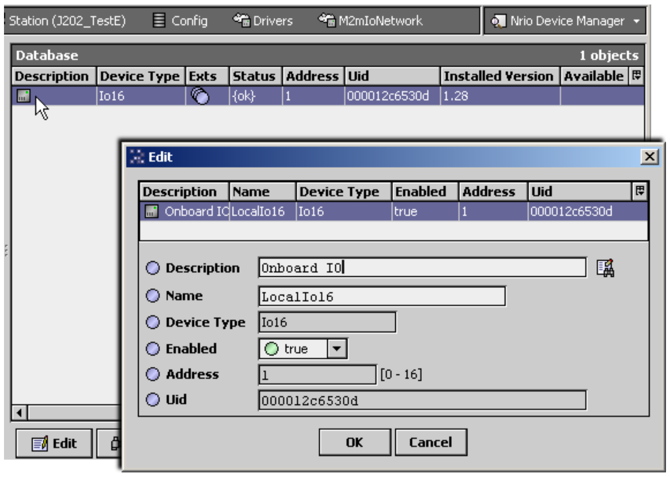
About IO processor updates to IoStatus

Each I/O processor scans analog inputs (UIs) every 45 ms, storing 6 consecutive readings in a buffer for each input. At this 270 ms cycle, the highest and lowest values in each buffer are discarded, and the four remaining values are averaged. Each UI's averaged value is then written into the concatenated IoStatus buffer that is read by the actrld process running on the JACE. This entire cycle repeats every 270 ms.

Nrio Device Manager

The default view on a M2mIoNetwork or NrioNetwork, the **Nrio Device Manager** provides a table based view of device level **NrioModule** components. If an M2mIoNetwork copied from the **nrio** palette, the only needed **NrioModule** is already configured, ready to access the onboard I/O of an M2M JACE.

Figure 5 M2mIoNetwork in **Nrio Device Manager**



Database 1 objects

Description	Device Type	Exts	Status	Address	Uid	Installed Version	Available
	Io16		{ok}	1	000012c6530d	1.28	

Edit

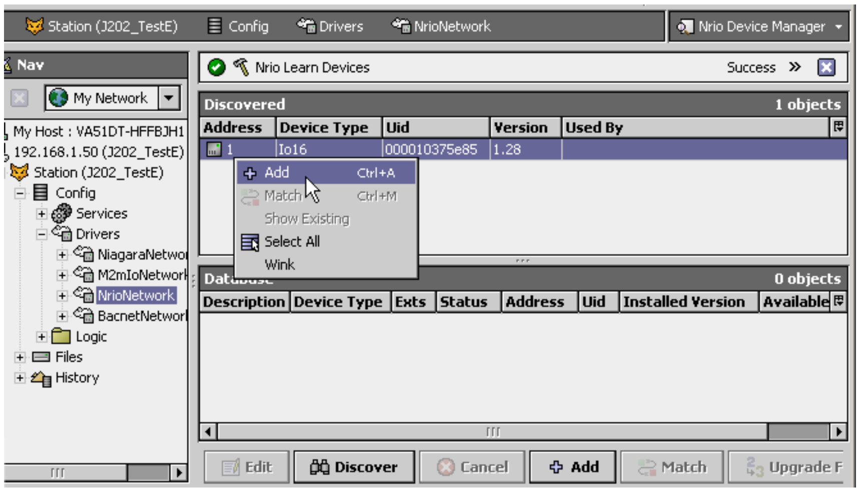
Description	Name	Device Type	Enabled	Address	Uid
Onboard IO	LocalIo16	Io16	true	1	000012c6530d

Description	Onboard IO
Name	LocalIo16
Device Type	Io16
Enabled	true
Address	1 [0 - 16]
Uid	000012c6530d

The default name for the child Nrio16Module is **LocalIo16**, and you can further edit by double-clicking and entering a Description value. This component auto discovers the onboard JACE IO module, and you can double-click on the Exts to go to its **Nrio Point Manager** view.

For an NrioNetwork, used to support remote I/O modules (devices available at some later date), the **Nrio Device Manager** also provides a Learn Mode to find such RS-485 connected devices. In this case, this view allows you to discover and add one or more device-level **NrioModule** components to the station database.

Figure 6 Adding NrioModule discovered using **Nrio Device Manager**



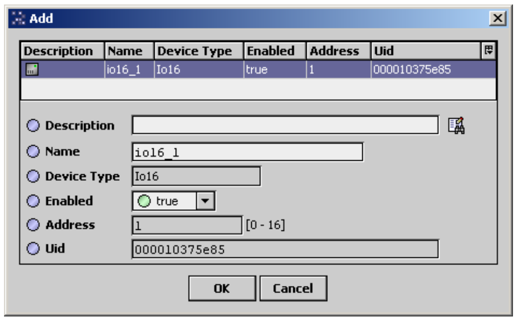
Discovered 1 objects

Address	Device Type	Uid	Version	Used By
1	Io16	000010375e85	1.28	

Database 0 objects

Description	Device Type	Exts	Status	Address	Uid	Installed Version	Available

Figure 7 Add dialog for NrioModule(s)



Description	Name	Device Type	Enabled	Address	Uid
	io16_1	Io16	true	1	000010375e85

Description	
Name	io16_1
Device Type	Io16
Enabled	true
Address	1 [0 - 16]
Uid	000010375e85

After adding all discovered NrioModule(s), you use the **Nrio Point Manager** view of each **NrioModule** to add Nrio proxy points—one for each available I/O hardware point (terminal address).

Nrio Device Manager usage notes

Note the following when using the **Nrio Device Manager**.

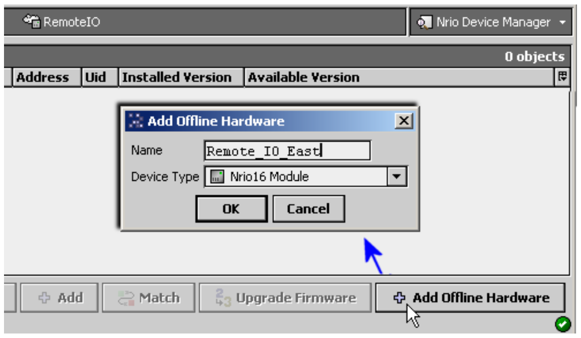
- For an M2mIoNetwork, only one child **Nrio16Module** is valid. This supports the onboard IO of an M2M JACE (JACE-202 Express).
- Currently, all Nrio-accessed IO is represented as "Io16", where each discovered **NrioModule** appears with a default name of **Io16_n**. Usually you change this name, except for the **NrioModule** under an M2mIoNetwork copied from the **nrio** palette—its default **LocalIo16** name is often acceptable. Note that any NrioModule's child **Points** extension dynamically reflects its parent's name.
- You may also enter a Description in the Add dialog when adding an **NrioModule**, or else do that in the **Edit** dialog later. Note that Address is read-only, and is automatically assigned by the online Discover (you cannot manually edit).
- Offline engineering is possible (using **Add Offline Hardware**), and can be useful when engineering is needed before being online with the exact IO device(s). In this case, when online with the IO devices, you use the Match feature in the **Nrio Device Manager**, following the Discover.

  This lets you match one-for-one a discovered **NrioModule** to the actual IO device. In the case of multiple external remote I/O modules (more than one), you typically use the Wink action on discovered devices before each Match, to correctly associate the software component to the right physical I/O module.

About Nrio offline engineering

The **Nrio Device Manager** has a **Add Offline Hardware** button that may be useful if engineering cannot wait until you are online with the actual IO. The following image shows the resulting dialog, in which you are prompted for name and device type. Note you must be online with the station to access this feature.

Figure 8 Add Offline Hardware in **Nrio Device Manager**



Name: Remote_IO_East
Device Type: Nrio16 Module

NOTE: The **`Device Type`** selection in the **Add Offline Hardware** dialog defaults to `Nrio16Module`; currently, this is the only valid selection.

Any **NrioModule** added offline will have value of zero (0) for both Address and Uid (Unique ID), along with a fault status. Its Fault Cause property will read "Invalid UID: Do Discover and Match.", which summarizes how to clear the fault (once online with the IO).

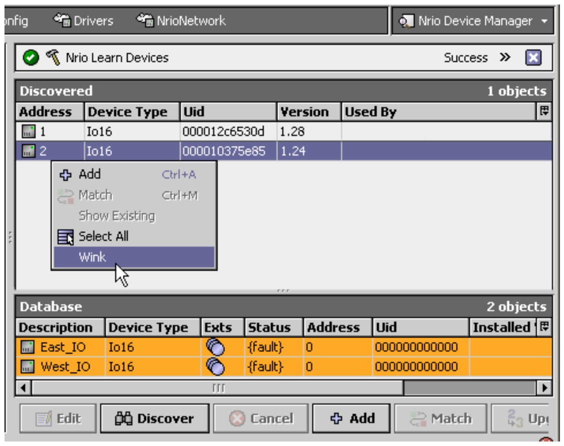
NOTE: Following adding an **NrioModule** using the **Add Offline Hardware** method, note you can still use Learn mode from the component's **Points** extension's **Nrio Point Manager** view to Discover and Add Nrio proxy points (all will have a fault status). And as needed, add other extension types (e.g. history) and link into control logic, etc.

However, such an **NrioModule** will not be operational until the JACE is online with the IO device, and only then after you do an online Discover and Match from the **Nrio Device Manager**. In the case of multiple Nrio-Modules on the NrioNetwork, you typically "wink" discovered IO devices before making each match.

About Nrio Wink

Discovered Nrio devices have an available right-click Wink action, as shown below.

Figure 9 Wink function is used to verify a discovered device



Discovered 1 objects

Address	Device Type	Uid	Version	Used By
1	Io16	000012c6530d	1.28	
2	Io16	000010375e85	1.24	

Database 2 objects

Description	Device Type	Exts	Status	Address	Uid	Installed
East_IO	Io16		{fault}	0	000000000000	
West_IO	Io16		{fault}	0	000000000000	

By default, the Wink command causes the selected I/O device to cycle its first digital output (relay output) On and Off for a period of 10 seconds. This can be used to confirm the device before matching it to a specific **NrioModule** component in the station database (typically, added using Add Offline Hardware).

NOTE: Along with other actions, each **NrioModule** component in the station also has the equivalent “Wink Device” action. In addition, each **NrioModule** component has two related properties: Wink Output and Wink Duration (allowing adjustment from first output (1) with a duration of 10 seconds). However, wink is typically used less after an **NrioModule** is added. In fact, immediately following successful configuration of an **NrioModule** (including its points) you may wish to hide this action on the **NrioModule**, to prevent future inadvertent cycling of the designated output. Do this from the slot sheet of any **NrioModule**, by setting the Hidden config flag on its winkDevice slot.

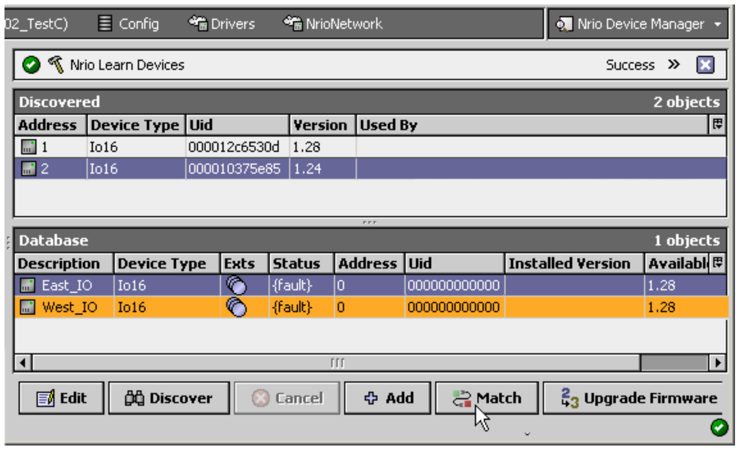
Following a device Wink to confirm the location of a device, typically you perform a match.

About Nrio Device Match

Match is a feature of Learn mode in the **Nrio Device Manager**, and is similar to device match available in the device manager view of some other drivers.

The following image shows a match about to be initiated between a discovered **Nrio16Module** and an offline-created **Nrio16Module**.

Figure 10 Match function is used to associate discovered device with offline-created NrioModule



Discovered 2 objects

Address	Device Type	Uid	Version	Used By
1	Io16	000012c6530d	1.28	
2	Io16	000010375e85	1.24	

Database 1 objects

Description	Device Type	Exts	Status	Address	Uid	Installed Version	Availabl
East_IO	Io16		{fault}	0	000000000000		1.28
West_IO	Io16		{fault}	0	000000000000		1.28

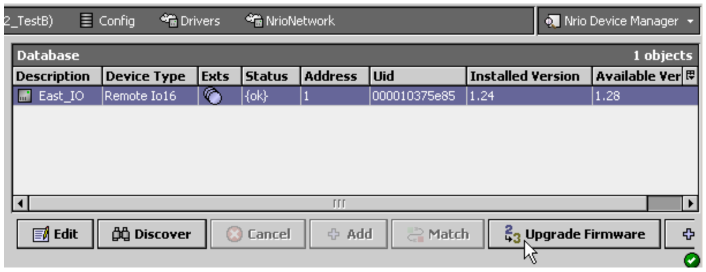
Following the match, the **NrioModule** uses the Uid and automatically-assigned Address of the selected discovered device, and its fault status is cleared. In the Discovered pane, the associated device becomes unavailable for selection (dimmed).

About Upgrade Firmware

The **Nrio Device Manager** has an **Upgrade Firmware** button. It is available when you select one or more NrioModules that have older IO processor firmware (than what is inside the JACE's **nrio** module).

NOTE: Control may be briefly affected during an IO firmware upgrade. Before starting a firmware upgrade, it is recommended to put associated controlled loads in a manually-controlled state. A firmware upgrade typically takes less than one minute for each NrioModule. After the firmware upgrade, controlled loads may be safely returned to system control. Also, ensure that power (and communications) to the JACE and IO is uninterrupted during the upgrade.

Figure 11 Upgrade Firmware in **Nrio Device Manager**



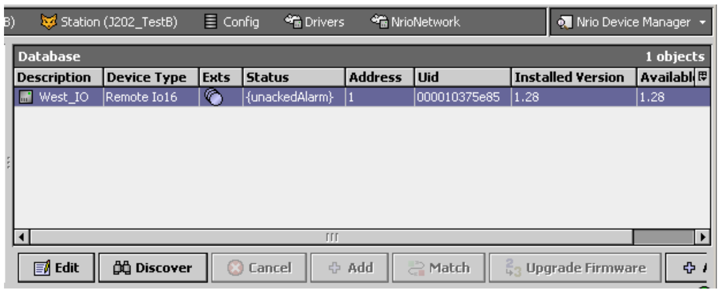
Database 1 objects

Description	Device Type	Exts	Status	Address	Uid	Installed Version	Available Ver
East_IO	Remote Io16		{ok}	1	000010375e85	1.24	1.28

When you click **Upgrade Firmware**, an associated job is sent to the station's JobService, where upon job competition you can see the step results. An **NrioModule** upgrade firmware job typically takes less than 1 minute, during which time the **NrioModule** briefly goes offline and the IO firmware file is downloaded from the JACE.

Following the upgrade, notice that the "installed version" and "available version" numbers match, and the **Upgrade Firmware** button is no longer available when the **NrioModule** is selected.

Figure 12 Following a firmware upgrade in **Nrio Device Manager**



Database 1 objects

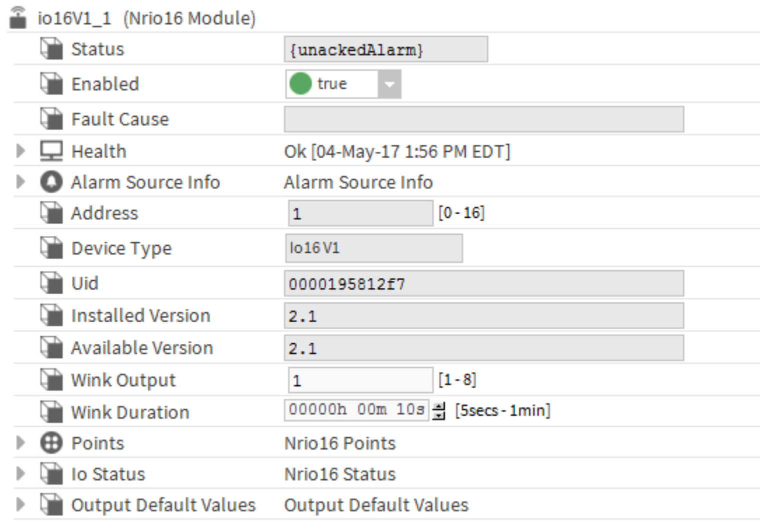
Description	Device Type	Exts	Status	Address	Uid	Installed Version	Availabl
West_IO	Remote Io16		{unackedAlarm}	1	000010375e85	1.28	1.28

Note also that device "Ping" alarms typically result from an IO firmware upgrade—from when the NrioModule(s) toggled offline and back online.

About the NrioModules

The Nrio modules, Nrio16Module and the added Nrio34Module (in Niagara 4.3 and later), represent an Nrio processor that services hardware I/O points. The host JACE controller communicates with Nrio processors using the device model of **NrioModule** (parent) to physical I/O points serviced by its processor as (child) Nrio proxy points.

Figure 13 **NrioModule** (Nrio16Module) property sheet


io16V1_1 (Nrio16 Module)

Status	{unackedAlarm}
Enabled	true
Fault Cause	
Health	Ok [04-May-17 1:56 PM EDT]
Alarm Source Info	Alarm Source Info
Address	1 [0 - 16]
Device Type	Io16 V1
Uid	0000195812f7
Installed Version	2.1
Available Version	2.1
Wink Output	1 [1 - 8]
Wink Duration	00000h 00m 10s [5secs - 1min]
Points	Nrio16 Points
Io Status	Nrio16 Status
Output Default Values	Output Default Values

A JACE with integral (onboard) Nrio-connected I/O, for example an M2M JACE, is represented by only one **NrioModule** (Nrio16Module), under its M2mIoNetwork (a specialized type of NrioNetwork). For a JACE with external I/O module(s) connected via RS-485, each RS-485 port with connected I/O modules requires its own associated NrioNetwork. Under such a network, one **NrioModule** represents each physical I/O device (one-to-one).

Under any Nrio network, each **Nrio16Module**, must have a unique `Address` property value from 1–16.

Similarly (in Niagara 4.3 and later), each **Nrio34Module** will require two of the available 16 address slots (1–16). This is because the Io34 is actually two controllers on one pc board. One is referred to as the primary controller, the other as the secondary. When an Io34 is viewed in the **Nrio Device Manager** the Address column shows the address of the primary controller, and the SecAddr column shows the address of the secondary controller.

For any type of NrioModule, the address value is automatically derived during an online Discover, and is a read-only value. Also during Discover, each I/O device's globally-unique "Uid" (Universal Identifier) property value is retrieved.

NOTE: The read-only six byte Uid is unique to each physical I/O device, assigned at manufacturing time. Future usage of Uid may incorporate bar code scanning and other automated methods.

Apart from the Enabled and Description properties, the wink-related properties are writable (`Wink Output`, `Wink Duration`), as well as the `Output Default Values` properties (Niagara 4.3 and later). All other **NrioModule** properties are read-only types.

NOTE: The **Nrio Device Manager** view of an NrioNetwork provides online Learn Mode discover and add commands to ensure all necessary NrioModules are created and configured properly for the host's hardware configuration.

Each **NrioModule** has a single important device extension: **Points** - The container for all its Nrio proxy points. Each proxy point represents a specific I/O terminal address, serviced by that board's I/O processor (s).

NrioModule properties

NrioModule properties can be categorized into two groups.

- Device status properties
- Config and I/O-related properties

  NOTE: As in other driver networks, the **NrioModule** has an available **Alarm Source Info** container slot you can use to differentiate **NrioModule** alarms from other component alarms in the station.

Device status properties

An **NrioModule** has typical device-level status properties:

- **`Status`**

  Status of NrioNetwork communications to this **NrioModule**. Possible status flags include:

  – ok - Normal communications, no other status flags.

  – disabled - **`Enabled`** property is set to `false`, either directly or in NrioNetwork. While status is disabled, all child Nrio points have disabled status; **NrioModule** polling is suspended.

  – fault - Typically an offline-added **NrioModule** (invalid 0 values for both Address and Uid).

  – down - Error communicating to the Nrio processor on I/O device—possibly an RS-485 communications problem if a remote I/O module.

- **`Enabled`**

  Either `true` (default) or `false`. Can be set directly or in parent NrioNetwork. See Status disabled description.

- **`Health`**

  Contains properties including timestamps of last "ok" time and last "fail" time, plus a string property describing last fail cause.

- **`Fault Cause`**

  If status has fault, describes the cause (e.g.: "Invalid UID: Do Discover and Match").

Config and I/O-related properties

In addition to common device-level Device status properties, **NrioModule** has the following unique properties, where most are read-only types unless noted.

Property	**Value**	**Description**
`Address`	0–16, 1 (default)	Each NrioModule under the NrioNetwork must have a unique "Address" property, an integer value from 1 to 16. Automatically derived (and associated) with physical I/O devices upon an online Discover.
`Device Type`	Io16, Io16V1, Io34	If in an M2mIoNetwork, this value is **Io16**. If in an NrioNetwork, this value is either Io16V1 or Io34, depending on the capability of the connected device.
`Uid`	000000000000	Unique ID. A six byte number, globally unique to this specific I/O hardware device. Automatically obtained from each device upon an online Discover.
`Installed Version`		Reflects the IO firmware revision installed in the I/O module/device.
`Available Version`		Reflects the IO firmware revision available in the JACE's installed **nrio** module. If this version number is later (higher) than the installed version number, you can initiate an I/O firmware upgrade from the **Nrio Device Manager**.
`Wink Output`	1–8, 1 (default)	(Writable) Specifies which digital output (relay output) is cycled On and Off when a Wink Device action is invoked on the **NrioModule**, where default is output 1. Note although the range is from 1 to 8, currently I/O hardware (onboard M2M JACE, 10-16-485) has only four relay outputs (1-4).

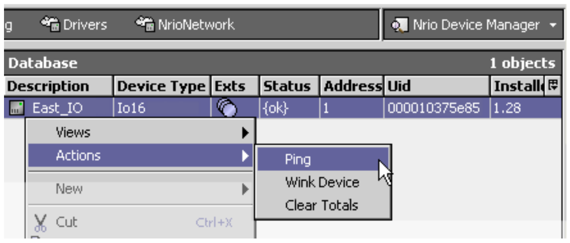
Property	Value	Description
`Wink Duration`	00000h 00m 10s (default) 5–60 seconds	(Writable) Specifies how long the wink output is cycled On and Off, at constant rate 1 second On, 1 second Off. Default value is 10 seconds, with a range from 5 seconds to 1 minute. **NOTE:** Wink is typically used only in the early stages of station configuration of NrioModules under an NrioNetwork. Following the competition of job engineering, you may wish to "hide" the Wink Device action on NrioModules, to prevent possible (inadvertent) unintended cycling of loads.
`LocalIo16` or `Io16_n`		(Nrio16Points) The **Points** extension/container for all Nrio proxy points for this I/O device. The name of this points extension reflects the name of the parent NrioModule component. Currently, no other properties exist. The default view is the **Nrio Point Manager**, which you use to add, modify, and delete child Nrio proxy points. These provide the interface to I/O hardware points.
`Io Status`	container	(NrioStatus) Contains a concatenated "Io Status" summary of current IO values in hexadecimal coded format, and numerous component children with individual hexadecimal values. **NOTE:** Typical usage of Io Status values are for advanced debug purposes only. This is the value last received by the actrld process running on the JACE.
`Output Default Values`	container	This property (in Niagara 4.3 and later) is a component that is used to specify the state of the relay and analog outputs when communications is lost and when communications is not established on power up. **NOTE:** By default, the outputs for the IO-16 will behave the same as in previous releases. You must modify the OutputDefaultValues property for each IO-16 to modify the behavior upon entering comm loss timeout or startup timeout states.
`Enable Comm Loss Defaults`	true/false (default)	If false, the DOs and AOs will be set to OFF and 0.0 vdc respectively, 8 seconds after communications is lost to the JACE. If true, each DO and AO will be set to the value specified by the user upon Comm Loss timeout. This timeout is also specified by the user.
`Enable Startup Defaults`	true/false (default)	If false, upon power-up the DOs and AOs will be set to OFF and 0.0 vdc respectively and remain until communications is established to the JACE. If true, upon power-up the DOs and AOs will be set to OFF and 0.0 vdc respectively and remain until: 1. Communications is established to the JACE or 2. The startup timeout expires in which case each DO and AO will be set to the value specified by the user.
`Comm Loss Timeout`	00s	Read-only comm loss timeout value (in seconds) which is specified in the NrioNetwork's Output Failsafe Config property
`Startup Timeout`	000s	Read-only comm loss timeout value (in seconds) startup timeout value which is specified in the NrioNetwork's Output Failsafe Config propery

Property	Value	Description
`DO1–DO4`	true/false (default)	Specifies the state for each DO when the unit enters the Comm Loss or Startup timeout state. **NOTE:** For the added Nrio34Module (in Niagara 4.3 and later) digital outputs number Do1–Do10.
`AO1–AO4`	0.0–10.0	This specifies the value for each AO when the unit enters the Comm Loss or Startup timeout state. **NOTE:** For the added Nrio34Module (in Niagara 4.3 and later) analog outputs number Ao1–Ao8.

NrioModule actions

Each **NrioModule** has three right-click actions, as shown for the Nrio16Module in the following image.

Figure 14 Actions for an Nrio16Module



Description	Device Type	Exts	Status	Address	Uid	Install
East_IO	Io16		{ok}	1	000010375e85	1.28

These actions are summarized as follows:

- Ping

  Sends an immediate ping message to the I/O device to verify “health”, identical to a periodic ping from the driver’s Monitor mechanism.

- Wink Device

  Causes a specific digital output (relay output) to cycle On and Off at a once per second rate for the specified wink duration, as configured by the NrioModule’s properties. It is generally recommended that you hide the winkDevice action after initial configuration—to prevent inadvertent load cycling.

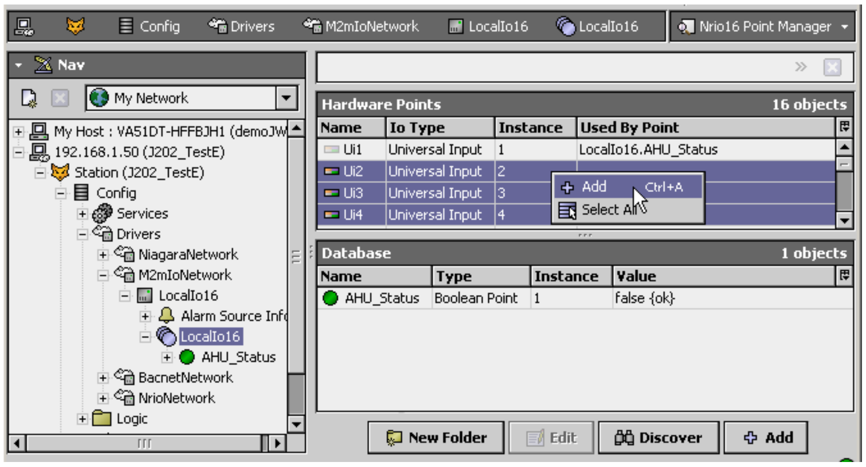
- Clear Totals

  Resets the accumulated Total value for all **CounterInputPoints** to zero (0), equivalent to invoking the “Reset” command on each point’s ProxyExt. Often, you may wish to hide this action to prevent inadvertent resets of point totals.

Nrio Point Manager

As the default view for the **Points** extension under an **NrioModule**, the **Nrio Point Manager** provides an online “Learn Mode” to find available I/O terminal points. You use this view to discover and add corresponding Nrio proxy points to the station database.

Figure 15 Adding Nrio proxy points discovered using Nrio16 Point Manager



Hardware Points 16 objects

Name	Io Type	Instance	Used By Point
Ui1	Universal Input	1	LocalIo16.AHU_Status
Ui2	Universal Input	2	
Ui3	Universal Input	3	
Ui4	Universal Input	4	

Database 1 objects

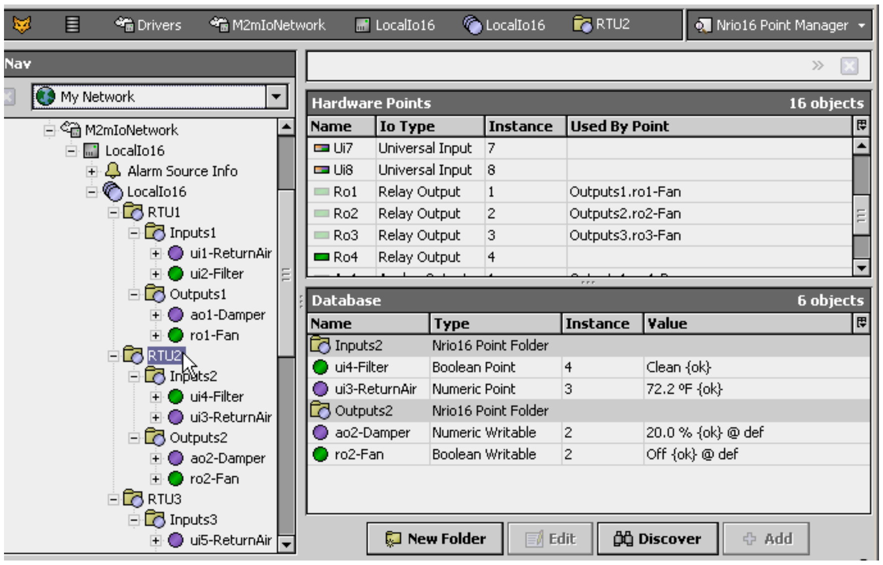
Name	Type	Instance	Value
AHU_Status	Boolean Point	1	false {ok}

Typically you add a proxy point for each discovered I/O terminal.

Point folder / discovery notes

Like many drivers, the **Nrio Point Manager** view provides a **New Folder** button to create point folders under the **Points** extension of the **NrioModule** device, where each point folder provides access to its own **Point Manger** view. The following image shows such an **Nrio Point Manager** view.

Figure 16 Nrio16 Point Manager view for an Nrio Point Folder



Hardware Points 16 objects

Name	Io Type	Instance	Used By Point
Ui7	Universal Input	7	
Ui8	Universal Input	8	
Ro1	Relay Output	1	Outputs1.ro1-Fan
Ro2	Relay Output	2	Outputs2.ro2-Fan
Ro3	Relay Output	3	Outputs3.ro3-Fan
Ro4	Relay Output	4	

Database 6 objects

Name	Type	Instance	Value
Inputs2	Nrio16 Point Folder		
ui4-Filter	Boolean Point	4	Clean {ok}
ui3-ReturnAir	Numeric Point	3	72.2 ºF {ok}
Outputs2	Nrio16 Point Folder		
ao2-Damper	Numeric Writable	2	20.0 % {ok} @ def
ro2-Fan	Boolean Writable	2	Off {ok} @ def

"Used By Point" shows name of proxy point. Also, right-click and select Show Existing to see full ord for point.

However, a few changes are unique to **Nrio Point Manager** views, summarized below.

- Discover (top) pane always shows availability of all points

  Regardless of the current hierarchy level of the **Nrio Point Manager** view, whenever in Learn mode, the top **Discover** pane always reflects the availability of all I/O hardware points. In other words, if you are in the manager view of a points folder `RTU2`, and some points on the controller are already mapped to proxy points under other point folders, they will show as unavailable (dimmed). This prevents creation of multiple proxy points for the same I/O terminal, which is invalid. Of course, you could manually duplicate an existing Nrio proxy point, but the duplicate point would have a "fault" status, along with an explanatory "Fault Cause" explanation.

Note that this **Discover** pane behavior is different than in most other drivers, where availability in the **Discover** pane applies to the current hierarchy level only (discovered items show available even if already proxied under another points folder branch of the **Points** container).

- Always All Descendants

  In any **Nrio Point Manager** view, the **Database** pane always lists all Nrio proxy points in that hierarchy level and lower, if applicable. In other words, all **Nrio Point Manager** views keep the "All Descendants" mode engaged (you cannot toggle it off).

  So, when looking at the top-level **Points** point manager view, all Nrio proxy points and `Nrio point` folders will be listed. If looking at the point manager view for Nrio points folder `RTU2`, any `Nrio point` folder children it has (say, Inputs2 and Outputs2) will be visible, along with all Nrio proxy points in all those `Nrio pointt` folders.

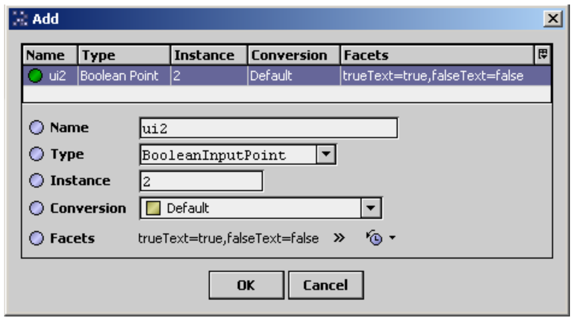
- Add discovered points to any selected point folder

  Any `Nrio point` folder shown in the **Database** pane of the **Nrio Point Manager** can be a target for adding discovered I/O points directly to it—without going to its own **Point Manager** view. In other words, click the target `Nrio point` folder to select/highlight it, then **Add** the discovered point(s) to create the proxy point(s) in that selected folder. Note that this behavior is also different than in most other drivers, where after adding, often you need to move points to different point folders.

Add and Edit dialog fields

When adding Nrio proxy points for discovered I/O points, the following items are available in the **Add** dialog box and afterwards, in the **Edit** dialog.

Figure 17 Add dialog in Nrio Point Manager



Name	Type	Instance	Conversion	Facets
ui2	Boolean Point	2	Default	trueText=true,falseText=false

Name: ui2
Type: BooleanInputPoint
Instance: 2
Conversion: Default
Facets: trueText=true,falseText=false

- **`Name`**

  Name of the Nrio proxy point (equivalent to right-click Rename, can be edited anytime). The default name appears as `AbbrevIoTypeTerminalNumber`, for example, `ui4`, `ao2`, and so on.

  Typically, you edit name from the default name to reflect the actual purpose of the I/O point, such as "Room101T," "AHU1_FanStatus," and so forth.

- **`Type`**

  The type of Nrio proxy point to create.

- **`Instance`**

  Corresponds to the I/O terminal address within that I/O type. Recommended to be left at default.

  NOTE: If an edit attempt is made to an instance already in use by another proxy point, the edit is discarded, and the previous instance value is retained.

- **`Conversion`**

The conversion type used between units in the proxy extension's "Device Facets" and the units in the parent point's Facets.

- **`Facets`**

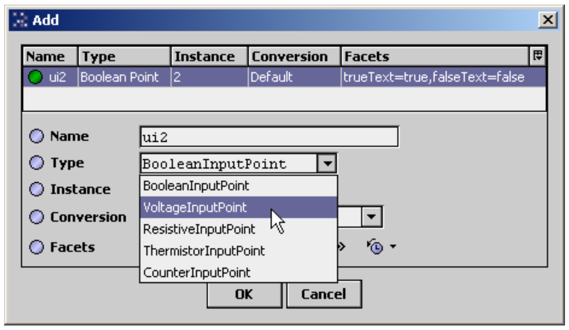
  Facets for the Nrio proxy point. Equivalent to accessing facets through the point's property sheet (and can be edited anytime).

Universal Input selection notes

All Nrio-capable JACE controllers and external I/O modules provide some number of universal input (UI) terminals. For example, the onboard I/O of an M2M JACE has eight UIs (terminals UI1—UI8). Unlike some other I/O devices where you must hardware configure each UI-type input using a jumper or switch, you do UI terminal configuration in software, from the **Nrio Point Manager**.

Do this at add time by selecting the needed input type in the **Add** dialog, as shown in the following image.

Figure 18 Select Nrio input type for each universal input



Name	Type	Instance	Conversion	Facets
ui2	Boolean Point	2	Default	trueText=true,falseText=false

Name: ui2
Type: BooleanInputPoint

For most I/O platforms, any of the following are valid universal input choices:

- VoltageInputPoint

  NumericPoint that reads a 0-to-10Vdc input signal and produces either a voltage value or linear scaled output value.

  NOTE: Select this also for use with a 4-to-20mA sensor, with a 500 ohm shunt resistor wired across the corresponding UI input terminals.

- ResistiveInputPoint

  NumericPoint that reads a resistive signal within a 0-to-100K ohm range and produces either a ohms value or linear scaled output value.

- ThermistorInputPoint

  NumericPoint that reads a Thermistor temperature sensor (Type 3, or other) signal and produces a scaled output value.

- CounterInputPoint

  NumericPoint that counts the number of contact closures at the input, and also calculates a rate value. One of these two values is configurable as the status numeric output value. Rate type is configurable as either fixed window, sliding window, or trigger type.

- BooleanInputPoint

  BooleanPoint that reads the current input as one of two boolean states (equipment status).

Any selection but BooleanInputPoint results in a standard NumericPoint, but with a different type Nrio input proxy extension. The BooleanInputPoint results in a BooleanPoint with an NrioBooleanInput proxy extension.

NOTE: After adding any Nrio proxy point, you can edit name, address, conversion, and facets if desired—but not type. To change type, you must delete the point and then add it again, selecting from the Type drop-down menu. The one exception here is the ResistiveInputPoint and ThermistorInputPoint, which are actually the same—except for the conversion type used in the ProxyExt.

Output type selection notes

Nrio-capable JACE controllers and external I/O modules typically have some number of digital outputs (DOs, typically relay type) and/or 0-to-10Vdc analog output (AO) terminals. For any discovered output I/O terminal, the Add dialog preselects the appropriate writable Nrio point type, as either:

- RelayOutputWritable

  A standard BooleanWritable point, but with an NrioRelayOutputWritable proxy extension.
- VoltageOutputWritable

  A standard NumericWritable point, but with NrioVoltageOutputWritable proxy extension.

Unlike when adding universal input points, there is no alternate selection of type available when adding output points.

About Nrio proxy points

Nrio proxy points are similar to other driver's proxy points. You can (and often do) add alarm and history extensions to them, and link them into other station logic as needed.

Nrio point types

There are 7 different Nrio point types, which derive from the following 4 control point types.

- BooleanPoint

  For boolean "universal input" type NrioBooleanPoint (BooleanInputPoint).
- NumericPoint

  For numeric "universal input" types NrioCounterPoint, NrioResistivePoint, NrioThermistorPoint, and NrioVoltagePoint. (CounterInputPoint, ResistiveInputPoint, ThermistorInputPoint, VoltageInputPoint)
- BooleanWritable

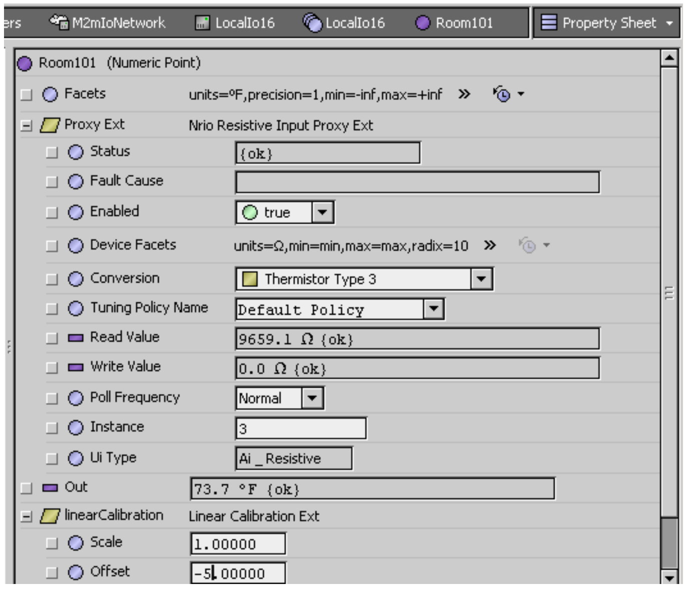
  For for boolean output type NrioRelayOutput (RelayOutputWritable).
- NumericWritable

  For numeric output type NrioVoltageOutput (VoltageOutputWritable).

Each Nrio point type is typically unique by one or more properties in its proxy extension. Also, the proxy extension in each type contains the same (common) properties. Things unique to each Nrio point type are explained in following sections, as well as common properties in each point's proxy extension.

Nrio Proxy Ext common properties

Figure 19 Common properties among Nrio proxy extensions



Room101 (Numeric Point)

Facets	units=°F,precision=1,min=-inf,max=+inf
Proxy Ext	Nrio Resistive Input Proxy Ext
Status	{ok}
Fault Cause	
Enabled	true
Device Facets	units=Ω,min=min,max=max,radix=10
Conversion	Thermistor Type 3
Tuning Policy Name	Default Policy
Read Value	9659.1 Ω {ok}
Write Value	0.0 Ω {ok}
Poll Frequency	Normal
Instance	3
Ui Type	Ai _Resistive
Out	73.7 °F {ok}
linearCalibration	Linear Calibration Ext
Scale	1.00000
Offset	-5 00000

As shown in the property sheet view, each Nrio point has the following properties in its proxy extension (Proxy Ext):

- **`Status`**

  (read only) Status of the proxy extension.

- **`Fault Cause`**

  (read only) If point has fault status, provides text description why.

- **`Enabled`**

  Either `true` (default) or `false`. While set to `false`, the point's status becomes disabled and Nrio polling is suspended.

- **`Device Facets`**

  (read only) Native facets used in proxy read or writes (Parent point's facets are used in point status display, and are available for edit in Add and Edit dialogs in the Point Manager).

- **`Conversion`**

  Specifies the conversion used between the "read value" (in Device Facets) and the parent point's output (in selected point facets).

- **`Tuning`**

  Assigned tuning policy, in Nrio is typically "Default Policy."

- **`Read Value`**

  (read only) Last value read, using device facets.

- **`Write Value`**

  (read only) Applies if writable point only. Last value written, using device facets.

- **`Poll Frequency`**

  Assigned poll frequency group, either Slow, Normal (default), or Fast.

- **`Instance`**

Point's I/O terminal address for its hardware type (UI, AO, DO), automatically found if added from Learn Mode discover. If Instance is set to out of range (relative to I/O board) or is a duplicate instance (same instance as same hardware type, same board), the point has a fault status.

- `UI Type`

  (read only) Nrio point type, such as "Resistive Input," "Boolean Output," and so on.

Conversion types in Nrio proxy points

In any Nrio proxy point a Conversion property is available. The options for this proper are described in the following sections.

NOTE: Currently, Workbench provides no filtering of Conversion types based on the Nrio proxy point being configured. For example, you see the same Conversion drop-down selection list for a VoltageInputPoint as you do for a RelayOutputWritable. However, only a few combinations are typically useful.

500 Ohm Shunt

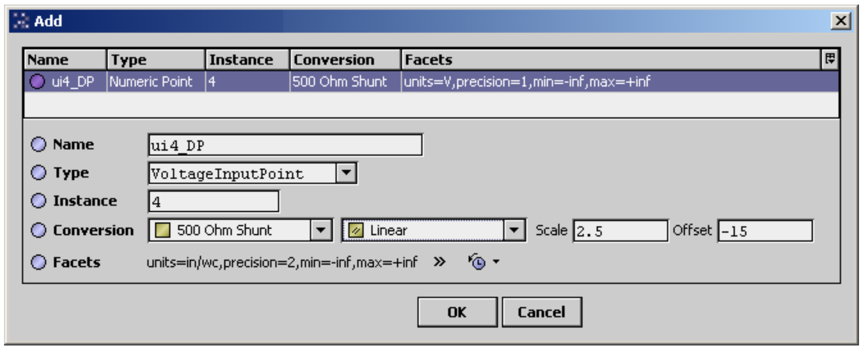
This Conversion type applies only to a VoltageInputPoint used to read a 4-to-20mA sensor, where the UI input requires a 500 ohm resistor wired across (shunting) the input terminals. The input signal is 2 to 10V. As compared to a Linear or Generic Tabular conversion, this selection provides better resolution near the upper (20mA/10V) range of the input, compensating for input clamping protection the circuitry automatically applies when input voltage rises above 3.9V.

NOTE: Currently, selection of `500 Ohm Shunt` produces a "secondary" selection for conversion type, with all conversion types shown again. However, only two secondary Conversion types are valid for 500 Ohm Shunt:

- Linear if the 4-20mA sensor has a linear response (typical).

  In the Scale and Offset fields for this Linear conversion, you enter whatever calculated values are needed to display in the appropriate units of measurement for that sensor. The following image shows an example configuration of such a point.

Figure 20 Example VoltageInputPoint used for 4-to-20mA sensor (Edit dialog from Point Manager)



Name	Type	Instance	Conversion	Facets
ui4_DP	Numeric Point	4	500 Ohm Shunt	units=V,precision=1,min=-inf,max=+inf

Name: ui4_DP
Type: VoltageInputPoint
Instance: 4
Conversion: 500 Ohm Shunt / Linear Scale 2.5 Offset -15
Facets: units=in/wc,precision=2,min=-inf,max=+inf

For background on this example configuration, see "Scale and offset calculation example 2".

- Generic Tabular if the 4-20mA sensor is non-linear in response. This is much less typical.

  In this case, you must enter (or import) a custom "source and result" .xml curve file, where the source (signal) values are between 0 to 10 (volts, where you multiply each mA value * 500), and the corresponding results are the measured values, in whatever units.

Default

(The default Conversion type). Conversion between "similar units" is automatically performed within the proxy point. For example, for a ThermistorInputPoint if you set the units in its facets to degrees F, the point output automatically converts to the appropriate value, from degrees C.

If you set the parent point's Facets to dissimilar units (often the case with a VoltageInputPoint), the parent point has a fault status to indicate a configuration error. In this case, you must select Linear conversion, and also set "Units" in the point's "LinearCalibrationExt" to match the point's facets.

Generic Tabular

(Requires **kitIo** module to be installed). This Conversion type allows non-linear support for devices other than for thermistor temperature sensors with units in temperature. Generic Tabular uses a lookup tablemethod similar to the "Thermistor Tabular" conversion, but without predefined output units.

Selection provides a popup **Tabular Conversion Dialog** in which you can enter a custom source-to-result non-linear curve, including the ability to import and export response curves.

Linear

This Conversion type applies to a VoltageInputPoint, ResistiveInputPoint, and VoltageOutputWritable points used with a linear-acting device.

For these points, you typically want the point's output value in some units other than Device Facets (voltage or resistance). The Linear selection provides two fields to make the transition:

- Scale: Determines linear response slope.
- Offset: Offset used in output calculation.

For related details, see "Scale and offset calculation (linear)".

NOTE: For a VoltageInputPoint used for a 4-to-20mA sensor (shunt resistor on UI input), another primary conversion type should be used: `500 Ohm Shunt`, with `Linear` selected as the "secondary" conversion.

Reverse Polarity

This Conversion type applies only to a BooleanInputPoint point or RelayOutputWritable, to reverse the logic of the hardware binary input or output.

NOTE: Be careful in the use of the reverse polarity conversion, as it may lead to later confusion when troubleshooting logic issues.

Tabular Thermistor

This Conversion type applies only to a ThermistorInputPoint point, where this selection provides a control for a popup dialog for a custom resistance-to-temperature value response curve, including ability to import and export response curve files.

Thermistor Type 3

This Conversion type applies only to a ThermistorInputPoint point, where it provides a "built-in" input resistance-to-temperature value response curve for Type 3 Thermistor temperature sensors.

Scale and offset calculation (linear)

Unless you want a VoltageInputPoint or VoltageOutputWritable to operate with facets (units) of volts, you must set the point's facets to the desired units, and change the Conversion property in its ProxyExt. Typically, you select a Conversion of Linear, except in the case of VoltageInputPoint used for a 4-20mA sensor, which requires a conversion of 500 Ohm Shunt, with Linear as the secondary conversion.

Linear conversion is also appropriate for a linearly-responding ResistiveInputPoint using facets other than ohms—although this may be less common than a non-linear application.

Linear requires you to enter your calculated scale and offset values in the Linear fields. Also, if an input point, you must set the "Units" in its Linear Calibration Ext to match an input point's facets—otherwise a fault status occurs.

Use the following formulas to calculate the Linear scale and offset values:

where: x1 and x2 are the Nrio values (e.g voltage), and y1 and y2 are the corresponding desired units.

- Scale

  Scale = (y2 - y1) / (x2 - x1)
- Offset

  Offset = y1 - (Scale * x1)

See example 1 and example 2.

Scale and offset calculation example 1

You have a 6-to-9Vdc actuator to control with a VoltageOutputWritable, in terms of 0-to-100% input. Configure this point's facets to have units: "misc," "percent" (%).

Set the ProxyExt's Conversion property to Linear, and enter these calculated scale and offset values:

- Scale

  Scale = (y2 - y1) / (x2 - x1)

  Scale = (100% - 0%) / (9V - 6V) = 100/3 = 33.3333333
- Offset

  Offset = y1 - (Scale * x1)

  Offset = 0% - (33.3333333 * 6V) = 0 - 200 = -200

Scale and offset calculation example 2

You have a linear 4-to-20mA differential pressure sensor to read with a VoltageInputPoint (using an external 500 ohm resistor wired across the input), reading effectively 2V to 10V. The range of this sensor is from -10 in.wc. to +10 in.wc. Configure this point's facets to have units: "pressure," "in/wc".

Set the ProxyExt's Conversion property to use "500 Ohm Conversion", then "Linear" as secondary, and in the Linear fields enter these calculated scale and offset values:

- Scale

  Scale = (y2 - y1) / (x2 - x1)

  Scale = (+10in.wc - -10in.wc) / (10V - 2V) = 20/8 = 2.5
- Offset

  Offset = y1 - (Scale * x1)

  Offset = -10in.wc - (2.5 * 2V) = -10 - 5 = -15

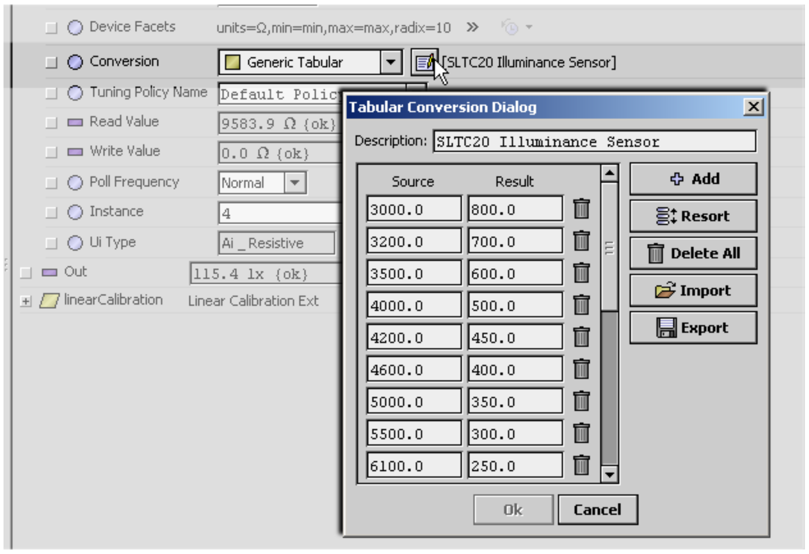
The following image shows this point's configuration in a Workbench dialog. Note that the Linear Calibration Ext for this point will require its "Units" property to be configured to match the facets for this point—otherwise, the point will have a fault status. See the "Linear Calibration Ext" section for related details.

Non-linear sensor support

Providing the **kitIo** module is installed, the conversion selection **Generic Tabular** provides a way for a VoltageInputPoint or ResistiveInputPoint to support a non-linear responding signal on a universal input (using point Facets other than temperature). Note that the ThermistorInputPoint also provides non-linear support for resistance-based temperature sensors, but point Facets are limited to temperature.

If you select Conversion type of "Generic Tabular" an edit control (note pad icon) appears in the property sheet beside it. Click the icon to see the **Tabular Conversion Dialog**, as shown in the following image.

Figure 21 Generic Tabular Dialog from edit control



Device Facets	units=Ω,min=min,max=max,radix=10
Conversion	Generic Tabular [SLTC20 Illuminance Sensor]
Tuning Policy Name	Default Polic
Read Value	9583.9 Ω {ok}
Write Value	0.0 Ω {ok}
Poll Frequency	Normal
Instance	4
Ui Type	Ai _ Resistive
Out	115.4 lx {ok}
linearCalibration	Linear Calibration Ext

Tabular Conversion Dialog

Description: SLTC20 Illuminance Sensor

Source	Result
3000.0	800.0
3200.0	700.0
3500.0	600.0
4000.0	500.0
4200.0	450.0
4600.0	400.0
5000.0	350.0
5500.0	300.0
6100.0	250.0

This dialog allows you to edit the current "source-to-results" non-linear curve used by the proxy point, import another curve (.xml file), or export (save) the current curve as an .xml file. In this way you can provide any custom non-linear curve needed. The following image shows a point with a non-linear curve described in the "Non-linear sensor example" topic.

The following notes apply to using this feature:

- A curve requires at least 2 points (rows), each using "Source" to "Results" values.
  - Source values use the "Device Facets" of the Nrio proxy point, and must be in the range of the controller input. Therefore, if a VoltageInputPoint, source values must be between 0 to 10 (Volts), or if a ResistanceInputPoint, source values must be between 0 to 100000 (Ohms).
  - Result values are typically the measured value, regardless of units, at each source point. This assumes no further scaling using a LinearCalibrationExt—i.e., its Scale property is the default "1".
- Points are used in ascending order, by the Source column.
- You can add as many points as is needed (there is no hard-coded limit). Click the **Add** button to add a new row for a point.
- If you skip a point, just add it at the end, and then click **Resort**. This automatically reorders all points in ascending order, by the Source column.
- Click the delete icon (trash can) beside any point to delete it from the curve.
  If you click the **Delete All** button, all points (plus any Description text) is removed (typically, you do this only to enter a new non-linear curve).
- Description text appears in the ProxyExt property sheet, beside the Conversion edit icon.
- When you click **Save**, the curve configuration is stored as part of the conversion object in the station (there is no association with any external file, in case you imported a non-linear curve).

  NOTE: If you update a non-linear curve (say, add a point) after it was imported in one or more Nrio points, you need to export it (save) and re-import it in other points, in order for them to be updated.

Non-linear sensor example

You have a photoresistor-based illuminance sensor that measures light output from 0 to 800 lux, supplying a resistance of 30K ohms to 3K ohms. Using a ResistiveInputPoint, select the conversion type **Generic Tabular**, and click the edit control for the **Tabular Conversion Dialog**.

In the dialog, enter the sensor's non-linear response curve (shown below). Configure this point's facets to have units: "illuminance," "lux" (lx), and the same units in the point's Linear Calibration Ext.

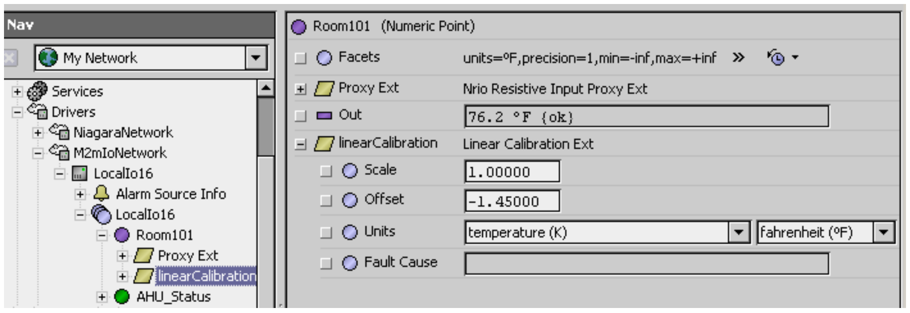
Table 2 Example illuminance sensor source (ohms) to results (lux)

Ohms	Lux
3000	800
3200	700
3500	600
4000	500
4200	450
4600	400
5000	350
5500	300
6100	250
6900	200
8200	150
10200	100
10900	90
11600	80
12400	70
13300	60
14400	50
15800	40
17700	30
20300	20
30000	0

Linear Calibration Ext

By default, three of the UI-type Nrio points (ResistiveInputPoint, ThermistorInputPoint, VoltageInputPoint) include a "linearCalibration" slot containing 4 properties, as shown in the following image.

Figure 22 Linear Calibration Extension



Room101 (Numeric Point)

Facets	units=°F,precision=1,min=-inf,max=+inf
Proxy Ext	Nrio Resistive Input Proxy Ext
Out	76.2 °F {ok}
linearCalibration	Linear Calibration Ext
Scale	1.00000
Offset	-1.45000
Units	temperature (K) fahrenheit (°F)
Fault Cause	

These properties allow you to “calibrate” the calculated value before is applied to the Out slot, where: [(calculatedValue x Scale) + Offset] = Out value.

Usage is optional, although Offset and Units are commonly configured.

NOTE: In most cases where the parent Nrio proxy point’s facets have been edited from defaults, note you must edit the Units value in the Linear Calibration Ext to match the units in the point facets, otherwise the parent proxy point will have a fault status!

Typically, you see this fault status immediately after you add a new input point, for example a VoltageInputPoint or ResistanceInputPoint, and configure it with a Linear conversion type (including a scale and offset), and then specify the point’s facets. It may not be immediately clear that the problem is in this Linear Calibration Ext, where you must match its Units value to the units in the point’s facets.

- Scale

  Default is 1.0. Typically, you leave this at default. One exception is if you copied the LinearCalibrationExt under a CounterInputPoint, in order to get a scaled total (see Note: below).

- Offset

  Default is 0.0. Can be either a positive or negative value, as needed. Often, this is useful to compensate for signal error introduced by sensor wiring resistance. If under a CounterInputPoint, leave at 0.

- Units

  Set this to the same units as in the parent proxy point’s facets (see Note: above).

- Fault Cause

  Provides a reason whenever this extension causes the parent proxy point to be in fault, e.g.: `Units between point and extension are not convertible.`

  You can also copy the linearCalibration extension without error to other Nrio points based on Numeric points. For example, a CounterInputPoint point if a “scaled total” out value is desired. Consider a CounterInputPoint for a flow rate meter that provides a contact closure for every 0.15 gallons, configured as:

- Facets: gallons [units: volume (m^3), gallon (gal)]
- NrioCounterInputProxyExt:
  - Conversion: Default
  - Output Select: Count
  - Rate Calc Type: FixedWindowRateType
  - Rate Calc:

    Scale: 9.0

    Interval: 15s
- LinearCalibrationExt:
  - Scale: 0.15

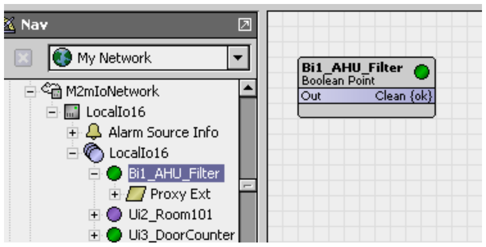
– Offset: 0.0

  By copying a LinearCalibrationExt to the CounterInputPoint and specifying the item quantity/pulse as the Scale value (in this case, 0.15), the count output value will read in gallons, rather than number of pulses. Note that the “rate calc” setup provides the “gallons per minute” scaling that will reflect in the Rate slot of the point’s ProxyExt, where Scale was calculated as 60 (sec/min) * 0.15 gal/pulse = 9.

BooleanInputPoint

A BooleanInputPoint is a BooleanPoint with NrioBooleanInputProxyExt. It configures a UI to read contact closures as a status boolean (equipment status, binary input, BI).

Figure 23 Nrio Boolean Input Point



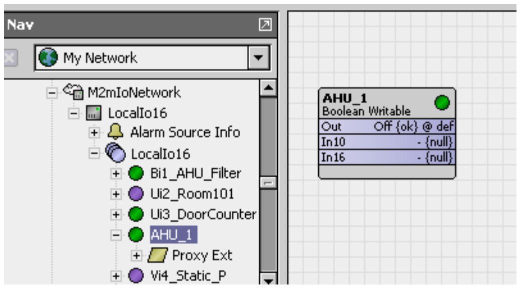
The BooleanInputPoint contains only common Nrio Proxy Ext properties. Among these, note that the “Device Facets” property is blank by default.

NOTE: The default operation is normal logic (closed contacts at UI is output value “true”). If needed, you can “flip” this logic in the ProxyExt by assigning a Conversion type of “Reverse Polarity.” This causes the “reversed” boolean state at the output value. Typically, you monitor normally open (N.O.) equipment contacts using normal logic.

RelayOutputWritable

A RelayOutputWritable is a BooleanWritable with NrioRelayOutputProxyExt. It represents a digital output (DO) as a relay type output. All standard BooleanWritable features apply, including right-click actions, priority input scheme, and minimum on/off properties.

Figure 24 Nrio Relay Output Writable



NOTE: The default operation is normal logic (closed contacts at DO if input value “true”). If needed, you can “flip” this logic in the ProxyExt by assigning a Conversion type of “Reverse Polarity.” This causes the “reversed” boolean state going from input to output.

The RelayOutputWritable has the common Nrio Proxy Ext properties.

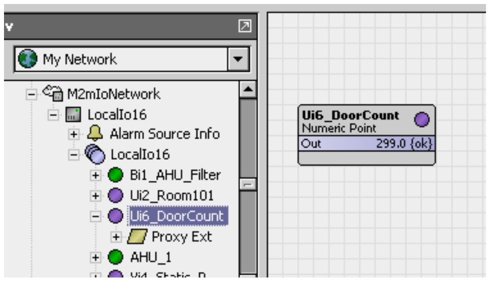
CounterInputPoint

A CounterInputPoint is a NumericPoint with NrioCounterInputProxyExt. It configures a UI to count dry-contact pulses up to 20 Hz, as well as calculate a numeric rate. In the ProxyExt, you specify which value is to appear at the proxy point’s Out slot (either Count or Rate) as a status numeric.

NOTE: The count value is maintained by the running JACE station, and not the I/O processor. This means any pulses received during periods of station shutdown or station startup are not recorded.

The proxy extension contains configuration properties for rate calculation, with different rate calculation methods available.

Figure 25 Nrio Counter Input Point



The following sections apply to the CounterInputPoint:

- About Nrio rate calculation
- Properties
- Actions

About Nrio rate calculation

Configuration properties of the CounterInputPoint's Proxy Ext specify which of three Nrio rate calculators are used, either:

All three have same purpose: to calculate a pulse rate and update the point's output to reflect the newly calculated value. Each calculator is triggered to perform in a slightly different way.

FixedWindowRateType

This rate calculator waits for the interval defined under the Rate Calc slot to elapse, then the recalculates the rate based on that interval.

Fixed window has 2 configurable rate calc values:

- Scale

  A multiplier to use for adjusting the reported output rate value. See "Properties", Rate Calc, Scale.
- Interval

  The amount of time between rate calculations.

SlidingWindowRateType

This rate calculator is similar to the fixed window rate calculator. It also calculates based on the specified interval. However, it performs the calculation every
interval/window seconds. This allows the rate to be updated more frequently while still maintaining that the calculation is based upon the specified interval. It then the recalculates the rate based on that interval.

Sliding window uses 3 configurable rate calc values:

- Scale

  A multiplier to use for adjusting the reported output rate value. See "Properties", Rate Calc, Scale.
- Interval

  The amount of time between rate calculations.
- Windows

The number of times during the interval that a recalculation is performed.

TriggerTypeRate

This rate calculator is the simplest. It simply adds a recalculateRate action to the parent point. Typically, you link the action to the output of a TriggerSchedule configured at a specific interval (e.g. at 0, 15, 30, and 45 minutes every hour). The calculator performs its recalculation based on the interval between now and last trigger.

Trigger type use only one configurable rate calc value:

- Scale

  A multiplier to use for adjusting the reported output rate value. See "Properties", Rate Calc, Scale.

Properties

NOTE: It is recommended to leave the Conversion at "Default" in the NrioCounterInputProxyExt, so as not to interfere with rate calculation. If a scaled Count output is needed, add a LinearCalibrationExt to the point, and enter the item quantity/pulse as the Scale value there. See "Linear Calibration Ext", including the ending Note: about this application.

In addition to common Nrio Proxy Ext properties, the following additional properties are available in the NrioCounterInputProxyExt:

- Output Select

  Specifies whether count total (Count) or count rate (Rate) is at the Out slot, as a status numeric. Default is Count.

- Total

  (read only) Total number of pulses counted since the Proxy Ext was last set or reset. Data type is Baja Long.

- Rate

  (read only) Calculated rate, based upon Rate Calc configuration. Data type is Baja Double.

- Rate Calc Type

  Selectable to one of three types provided in the Nrio module:

  – FixedWindowRateType (default)

  – SlidingWindowRateType

  – TriggerRateType

    NOTE: Future rate calculators may be developed by third-parties. When this occurs, they will be available (if that module is installed), as pull-down menu options other than Nrio.

- Rate Calc

  Contains from one to three properties used in rate calculation, according to the selected rate calc type (see "About Nrio rate calculation"):

  – Scale

    Default is value is 1. Appropriate value depends on the item quantity/pulse and desired rate units. Two examples are provided:

    Power Meter

    Each meter pulse indicates 0.15 kWh, and desired rate units is kW.

    Scale = 3600 (sec/hr) * 0.15 kWh/pulse = 54

    Liquid Flow Meter

Each meter pulse indicates 0.375 liters, and desired rate units is liters/minute (L/min).

Scale = 60 (sec/min) * 0.375 liters/pulse = 22.5

– Interval

(not available if TriggerRateType) Default is value is 1 minute.

– Windows

(available only if SlidingWindowRateType) Default is value is 6.

- Rate Calc Time

  (read only) Reflects timestamp of last rate calculation.

Actions

The NrioCounterInputProxyExt has the following available actions:

- Reset

  Sets the point's Total (and Out value, if outputSelect is Count) to zero (0).
- Set

  Sets the point's Total (and Out value, if outputSelect is Count) to a specified count. Note this is an un-scaled (actual number of pulses) value, so if a scaled Count is configured, it will show differently.

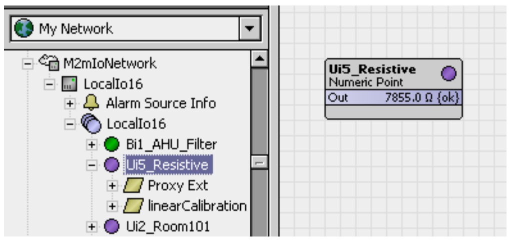
ResistiveInputPoint

A ResistiveInputPoint is a NumericPoint with NrioResistiveInputProxyExt. It configures a UI to read a resistance from 0 to 100,000 ohms. Default configuration provides ohms. If other units are needed, select conversion type of either Linear (for linear response) or Generic Tabular, if non-linear.

NOTE: Universal inputs (UIs) are optimized for resistive temperature sensors in 10K ohm range. Any temperature sensor far outside this range, for example a 100 ohm or 1K ohm type, will have poor resolution—and thus be unusable! In this case, install a compatible transmitter that provides a linear 0-to-10V or 4-to-20mA signal, and use a VoltageInputPoint instead.

The ResistiveInputPoint is pre-configured with a linearCalibration extension, to allow for offset correction.

Figure 26 Nrio Resistive Input Point



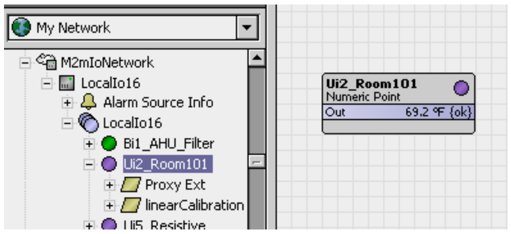
The ResistiveInputPoint's Proxy Ext contains only common Nrio Proxy Ext properties. Among these, note that the read-only "Device Facets" property is preset to ohms.

ThermistorInputPoint

A ThermistorInputPoint is a NumericPoint with NrioResistiveInputProxyExt configured for a UI to read a thermistor temperature sensor (10K ohm type). It is pre-configured with a linearCalibration extension for offset correction.

NOTE: A ThermistorInputPoint is the same as a ResistiveInputPoint, only with a different “Conversion” setting in its ProxyExt. However, the Nrio Point Manager lists this point separately, so it is covered here separately.

Figure 27 Nrio Thermistor Input Point



The point’s Proxy Ext contains common Nrio Proxy Ext properties. Among those properties, note that the read-only “Device Facets” is preset to ohms.

Depending on the type of thermistor temperature sensor you are using, you can select a Conversion type of either Type-3 Thermistor, or any other using a Tabular Thermistor conversion.

Type-3 Thermistor

Table 3-3 shows the hard-coded response curve used if the ProxyExt property Conversion is set to “Thermistor Type 3”:

Type-3 Thermistor Nrio Curve

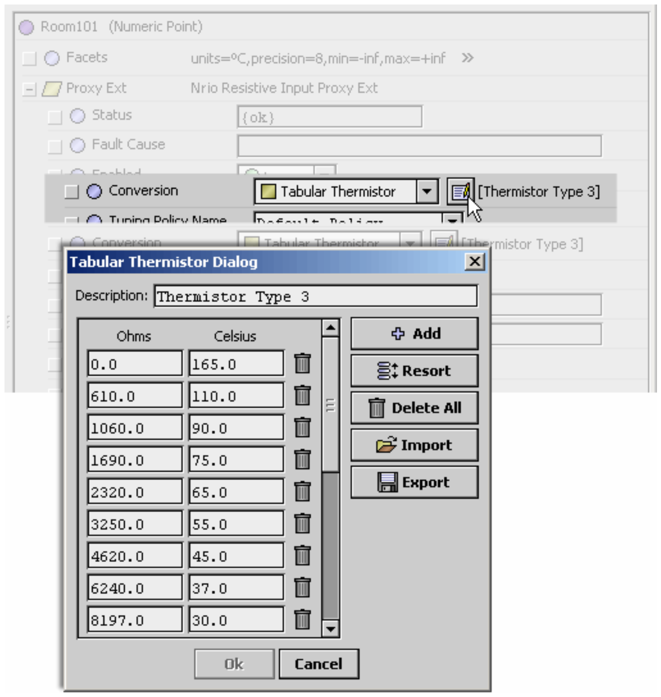
Ohms	Deg. C
0	165
610	110
1060	90
1690	75
2320	65
3250	55
4620	45
6240	37
8197	30
10000	25
12268	20
15136	15
18787	10
23462	5
29462	0
37462	-5

Ohms	Deg. C
47549	-10
61030	-15
78930	-20
100000	-25

Tabular Thermistor notes

In the ProxyExt of the thermistor point, if you select Conversion type of "Tabular Thermistor," an edit control (note pad icon) appears in the property sheet beside it. Click the icon to see the **Tabular Thermistor** dialog, as shown below.

Figure 28 Tabular Thermistor Dialog from edit control



Room101 (Numeric Point)
Facets units=°C,precision=8,min=-inf,max=+inf
Proxy Ext Nrio Resistive Input Proxy Ext
Status {ok}
Fault Cause
Conversion Tabular Thermistor [Thermistor Type 3]

Tabular Thermistor Dialog
Description: Thermistor Type 3

Ohms	Celsius
0.0	165.0
610.0	110.0
1060.0	90.0
1690.0	75.0
2320.0	65.0
3250.0	55.0
4620.0	45.0
6240.0	37.0
8197.0	30.0

This dialog allows you to edit the current ohms-to-degrees Celsius curve used by the proxy point, import another thermistor curve (.xml file), or export (save) the current thermistor curve as an .xml file. In this way you can provide any custom thermistor curve needed.

The following notes apply to using this feature:

- A curve requires at least 2 points (rows), each using ohms to degrees Celsius values.
- Points are used in ascending order, by the Ohms column.
- You can add as many points as is needed (there is no hard-coded limit). Click the **Add** button to add a new row for a point.
- If you skip a point, just add it at the end, and then click **Resort**. This automatically reorders all points in ascending order, by the Ohms column.

- Click the delete icon (trash can) beside any point to delete it from the curve.
  If you click the **Delete All** button, all points (plus any Description text) is removed (typically, you do this only to enter a new thermistor curve).
- Description text appears in the ProxyExt property sheet, beside the Conversion edit icon.
- When you click **Save**, the curve configuration is stored as part of the conversion object in the station (there is no association with any external file, in case you imported a thermistor curve.

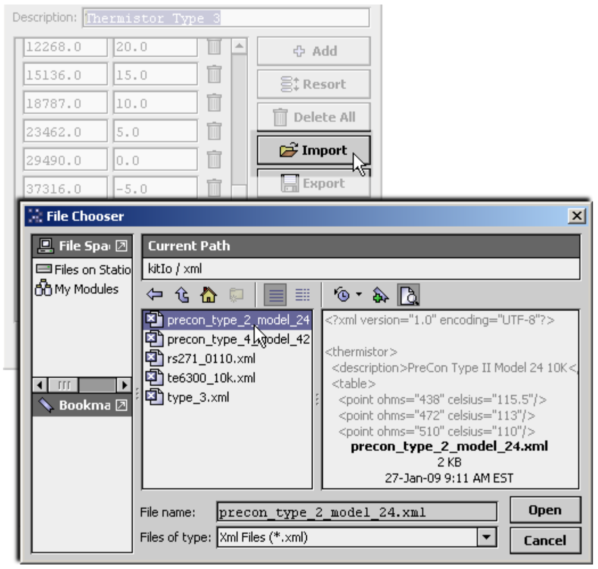
  NOTE: If you update a thermistor curve (say, add a point) after importing it into one or more Nrio points, you need to import it in those points again, in order for them to be updated.

Curve File Import/Export notes

As needed, import (load) or export (save) a thermistor or non-linear curve file in xml format using the **Import** and **Export** buttons in the either the **Tabular Thermistor Dialog**) for a "Tabular Thermistor" conversion, or **Generic Tabular Dialog** for a "Generic Tabular" conversion.

When you do this, the standard Niagara **File Chooser** dialog appears, as shown below.

Figure 29 File Chooser to locate thermistor .xml file for Import



Description: Thermistor Type 3

12268.0	20.0
15136.0	15.0
18787.0	10.0
23462.0	5.0
29490.0	0.0
37316.0	-5.0

Current Path
kitIo / xml

precon_type_2_model_24
precon_type_4_model_42
rs271_0110.xml
te6300_10k.xml
type_3.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<thermistor>
 <description>PreCon Type II Model 24 10K<
 <table>
  <point ohms="438" celsius="115.5"/>
  <point ohms="472" celsius="113"/>
  <point ohms="510" celsius="110"/>
```

precon_type_2_model_24.xml
2 KB
27-Jan-09 9:11 AM EST

File name: precon_type_2_model_24.xml
Files of type: Xml Files (*.xml)

To import a thermistor curve file, navigate to the xml folder in the **kitIo** module (Jar) file in your modules (My Modules). Or, navigate to any other location you have a thermistor curve file. Import any other saved, custom non-linear conversion .xml files the same way.

NOTE: The source .xml file for importing a thermistor or non-linear curve is not required to be in any installed module, either on the target JACE or your Workbench PC. However, as a convenience, a few standard thermistor curve files are included in the **kitIo** module. Additional ones may be added in future builds.

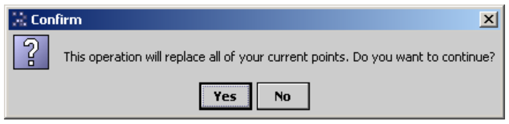
As of the date of this document, the **kitIo** module contains an xml folder with five (5) different thermistor curves in xml format, as follows:

- precon_type_2_model_24.xml — For Precon type 2 model 24 sensor
- precon_type_4_model_42.xml — For Precon type 4 model 42 sensor
- rs271_0110.xml — For Radio Shack sensor model 271-0110
- te6300_10k — For TE-6300 10K type sensor
- type_3.xml — For standard Type-3 Thermistor sensor

Upon any curve file import, all existing thermistor or non-linear curve configuration for that Nrio point is overwritten (replaced by configuration in the imported file). A confirmation dialog advises you of this before you import.

Figure 30 Curve file import confirmation

NOTE:


Confirm
This operation will replace all of your current points. Do you want to continue?
Yes
No

To export a curve .xml file, use the **File Chooser** to navigate to any location you wish to save the current thermistor or non-linear curve configuration, and supply a File Name. You can use this saved .xml file in the configuration of any other like Nrio proxy point, whether in this station or another station.

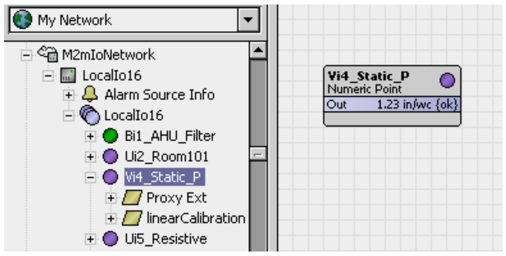
VoltageInputPoint

A VoltageInputPoint is a NumericPoint with NrioVoltageInputProxyExt. It configures a UI to read a Vdc signal from 0-to-10V. Currently, configuration provides for reading in volts or a linear response in other units, by selecting "Linear" as the conversion type.

NOTE: Besides reading a 0-10Vdc sensor, this point is also used for a 4-20mA sensor, where an external 500 ohm resistor is wired across the UI terminals. In this case only, select the "Shunt500 Ohm Conversion" as the point's conversion type.

The VoltageInputPoint is pre-configured with a linearCalibration extension, to allow for offset correction.

Figure 31 Nrio Voltage Input Point

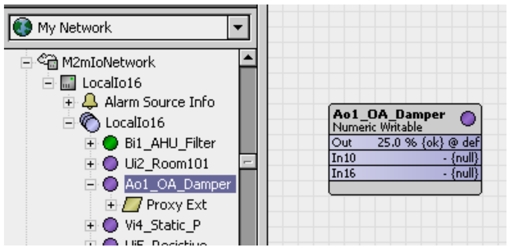


The VoltageInputPoint's Proxy Ext contains only common Nrio Proxy Ext properties. Among these, note that the read-only "Device Facets" property is preset to volts (V).

VoltageOutputWritable

A VoltageOutputWritable is a NumericWritable with NrioVoltageOutputProxyExt. It represents a 0-to-10Vdc analog output (AO). All standard NumericWritable features apply, including right-click actions and priority input scheme.

Figure 32 Nrio Voltage Output Writable



The point's Proxy Ext contains common Nrio Proxy Ext properties. Among those properties, note that the read-only "Device Facets" is set to units Volts (V). For operation in units other than volts, see "Scale and offset calculation (linear)".

In addition to the common Nrio Proxy Ext properties, the following additional properties are also available under a **Hardware Override** container slot.

NOTE: The following four Hardware Override properties were intended for a I/O hardware module that was never supported. Therefore, they are not functional and can be safely ignored.

- Overridden

  (read only, transient) Either false or true, as to whether the hardware override switch is active. Uses Baja data type Boolean.

- Fault Enable

  Either false or true (default). Enables a hardware override to mark the parent point in fault (while overridden is true).

- Alarm Enable

  Either false (default) or true. Enables a hardware override to mark the parent point in alarm (while overridden is true).

  If you configure both Fault Enable and Alarm Enable true, a fault alarm is generated upon an active hardware override.

- Alarm Class

  The alarm class to use for a hardware override-generated alarm or fault.

Chapter 3 Nrio Plugin Guides

Topics covered in this chapter

Plugins provide views of components, and can be accessed many ways—for example, double-click a component in the tree for its default view. In addition, you can right-click a component, and select from its **Views** menu. For summary documentation on any view, select **Help→ On View** (F1) from the Workbench menu, or press F1 while the view is open.

nrio-NrioDeviceManager

Use the **Nrio Device Manager** to add, edit, and access NrioModules that represent physical I/O on the host JACE controller, and/or external I/O modules connected to a specific RS-485 port of the JACE. The Nrio Device Manager is a view on the M2mIoNetwork or NrioNetwork. To view, right-click the network and select **Views→ Nrio Device Manager**, or simply double-click the network.

Columns in the **Discovered** and **Database** panes of this view are summarized below.

Discovered

The **Discovered** pane of this view has the following available columns:

- Address

  Reflects unique module address (1-16) under this Nrio network, automatically assigned during an online Discover. For the Nrio34Module in Niagara 4.3 and later, this is the address for the primary controller in the module.

  NOTE: An M2mIoNetwork supports only a single Nrio16Module—it always has 1 for Address.
- SecAddr

  In Niagara 4.3 and later, reflects unique module address for the secondary controller in a Nrio34Module.
- DeviceType

  Type of I/O represented by the discovered IO device (currently there is only **Io16**)
- Uid

  Globally Unique ID, as 6-byte hexadecimal number, for the associated I/O hardware represented by this NrioModule. It is automatically acquired during an online Discover.
- Versions

  Indicates the IO firmware revision currently installed in this IO hardware.
- Used By

  Has a value only if the IO device is already represented in the station's database, which is the name of the associated NrioModule.

Database

The **Database** pane of this view has the following available columns:

- Description

  Text descriptor for this module of I/O.

- Name

  Component name of the NrioModule.
- DeviceType

  Type of I/O represented by the NrioModule (currently there is only **Io16**)
- Exts

  Shortcut to the device extensions (only one extension, for Nrio proxy points).
- Status

  Current status for the NrioModule.
- Health

  Current health and last Ok (or Fail) timestamp for the NrioModule.
- Enabled

  Whether the NrioModule is enabled (true) or not (false).
- Address

  Reflects unique module address (1-16) under this Nrio network, automatically assigned during an online Discover. A "0" value means the NrioModule was added using **Add Offline Hardware** feature, or else copied from nrio palette. For the Nrio34Module in Niagara 4.3 and later, this is the address for the primary controller in the module.
- SecAddr

  In Niagara 4.3 and later, reflects unique module address (1-16) for the secondary controller in an Nrio34-Module under this Nrio network, automatically assigned during an online Discover. A "0" value means the NrioModule was added using **Add Offline Hardware** feature, or else copied from nrio palette..
- Uid

  Globally Unique ID, as 6-byte hexadecimal number, for the associated I/O hardware represented by this NrioModule. It is automatically acquired during an online Discover. Will be all zeros (0) if the NrioModule was added using **Add Offline Hardware** feature, or else copied from nrio palette.
- Installed Versions

  Indicates the IO firmware revision currently installed in this IO hardware.
- Available Version

  Indicates the IO firmware revision in the JACE's nrio module, available for download to one or more selected NrioModules.

nrio-Nrio16PointManager

Use the **Nrio16 Point Manager** to add, edit, and access Nrio proxy points under the Nrio16Points extension of a selected Nrio16Module (or points extension of any NrioModule), or in an `Nrio16PointFolder`. The **Nrio16 Point Manager** is the default view on both types of these components. To view, right-click the Nrio16Points extension or `Nrio16PointFolder` and select **Views→ Nrio Point Manager**, or simply double-click the component.

Columns in the **Discovered** and **Database** panes of this view are summarized below.

Discovered

The **Discovered** pane of this view has the following available columns:

- Name

Shows a default name for each available IO point, using format `AbbrevIoTypeTerminalNumber`, for example, `ui4`, `ao2`, and so on.

- Io Type

  Type of I/O for each point, such as `Universal Input`, `Relay Output`, or `Analog Output`.

- Instance

  Equals the hardware I/O terminal number for that IO type, for example 1 for UI 1 or AO 1.

- Used By

  Has a value only if the IO point is already represented in the station's database, formatted as `ParentName.NrioProxyPointName`. For example, `Onboard_IO.AHU_status` or `Inputs1.ReturnAir`

  NOTE: To see the complete ord for the associated proxy point, right-click and select "Show Existing".

Database

The **Database** pane of this view has the following available columns:

- Name

  Component name of the Nrio proxy point.

- Type

  Base control point type, such as BooleanPoint, NumericPoint, NumericWritable.

- Instance

  Equals the hardware I/O terminal number for that IO type, for example 1 for UI 1 or AO 1.

- Ui Type

  Descriptor for type of UI (if applicable), blank if point is an AO or DO. For example, `DI_Normal`, `AI_Resistive`, `AI_0to10_Vdc`, `DI_High Speed`, and so on.

- Conversion

  Conversion type used in Nrio ProxyExt, such as `Linear`, `Default`, `500 Ohm Shunt`, `Thermistor Type3`, and so on.

- Value

  Current out value and status of proxy point.

- Facets

  Facets information string for proxy point, including units, precision, and min/max value for numeric points, or true/falseText for boolean points.

Nrio Plugin Guides Summary

Summary information is provided on views specific to components in the `nrio` module, as follows.

- NrioDeviceManager
- Nrio16PointManager

Chapter 4 Nrio Component Guides

Topics covered in this chapter

These component guides provides summary help on Ndio components.

Summary information is provided on components in the **nrio** module.

nrio-FixedWindowRateType

This topic contains a short description of a component or the component plugin view.

Contains rate calculation parameters for the parent NrioCounterInputProxyExt.

nrio-LinearCalibrationExt

This topic contains a short description of a component or the component plugin view.

LinearCalibrationExt is an point extension that allows for calibration adjustment of an input to a known measured value.

nrio-M2mIoNetwork

This topic contains a short description of a component or the component plugin view.

The M2mIoNetwork represents the onboard I/O of a M2M JACE, or JACE-x02 Express. This network is a specialized version of the NrioNetwork, where properties Port, Trunk, and Baud Rate are read-only with values used to communicate to the JACE's onboard I/O processor. As with an NrioNetwork, the **NrioDeviceManager** is the default view.

nrio-Nrio16Module

This topic contains a short description of a component or the component plugin view.

Nrio16Module represents either the onboard I/O of a JACE or an external I/0 module (T-IO-16-485T-IO-16-485IO–R–16). You use the **Nrio16PointManager** view of its child Nrio16Points extension to discover, add, and edit Nrio proxy points.

nrio-Nrio34Module

This topic contains a short description of a component or the component plugin view.

Nrio34Module represents an external I/0 module (IO–R–34). You use the **Nrio16PointManager** view of its child Nrio16Points extension to discover, add, and edit Nrio proxy points.

nrio-Nrio16Points

This topic contains a short description of a component or the component plugin view.

Nrio16Points (name is same as parent NrioModule's) is the **Points** container for Nrio proxy points under a Nrio16Module type of **NrioModule**. As with any **NrioModule**, the default and primary view for the Nrio16-Points container is the **Nrio16PointManager**. Apart from this view, this slot serves only as the top container for all Nrio proxy points under that **NrioModule** (it has no other properties). This component is a frozen slot on any **NrioModule**.

nrio-NrioBooleanInputProxyExt

This topic contains a short description of a component or the component plugin view.

NrioBooleanInputProxyExt is the proxy extension for a BooleanInputPoint. It represents an Nrio boolean universal input.

nrio-NrioCounterInputProxyExt

This topic contains a short description of a component or the component plugin view.

NrioCounterInputProxyExt is the proxy extension for a CounterInputPoint. It represents an Nrio universal input configured for pulse-count and rate determination from dry contacts. Rate configuration is selectable as either:

- Fixed window type
- Sliding window type
- Trigger type

Additional rate parameters must be entered in the selected child component, such as FixedWindowRateType.

nrio-NrioNetwork

This topic contains a short description of a component or the component plugin view.

NrioNetwork represents an RS-485 connection to one or more remote I/O modules (NrioModules). A JACE with onboard I/O requires one M2mIoNetwork for that I/O, and if additional remote I/O modules are connected to an RS-485 port, a separate NrioNetwork for each such RS-485 port. As with the M2mIoNetwork, the NrioDeviceManager is the default view on any NrioNetwork.

nrio-Nrio16PointFolder

This topic contains a short description of a component or the component plugin view.

`Nrio16PointFolder` is an optional container for Nrio proxy points under the Nrio16Points extension of an Nrio16Module. The `Nrio16PointFolder` is available in the **Nrio Point Manager** via the **New Folder** button.

nrio-NrioPollScheduler

This topic contains a short description of a component or the component plugin view.

The NrioPollScheduler is the Nrio implementation of a poll scheduler in a NrioNetwork. Operation is similar to most other polling networks, except that polling picks values from each IO device's IoStatus slot, rather than result in discrete poll messages to IO devices.

nrio-NrioRelayOutputProxyExt

This topic contains a short description of a component or the component plugin view.

NrioRelayOutputProxyExt is the proxy extension for a RelayOutputWritable. It represents an Nrio boolean output.

nrio-NrioResistiveInputProxyExt

This topic contains a short description of a component or the component plugin view.

NrioResistiveInputProxyExt is the proxy extension for a ResistiveInputPoint. It represents an Nrio universal input that reads a resistance (ohms) signal or a Thermistor temperature sensor.

nrio-NrioVoltageInputProxyExt

This topic contains a short description of a component or the component plugin view.

NrioVoltageInputProxyExt is the proxy extension for a VoltageInputPoint. It represents an Nrio universal input that reads a 0-to-10Vdc input signal, or a 4-to-20mA signal (with a 500 ohm resistor wired across the input terminals).

nrio-NrioVoltageOutputProxyExt

This topic contains a short description of a component or the component plugin view.

NrioVoltageOutputProxyExt is the proxy extension for an VoltageOutputWritable. It represents an Nrio analog output (AO) that produces 0-to-10Vdc.

nrio-SlidingWindowRateType

This topic contains a short description of a component or the component plugin view.

Contains rate calculation parameters for the parent NrioCounterInputProxyExt.

nrio-ThermistorType3Type

This topic contains a short description of a component or the component plugin view.

ThermistorType3Type is a thermistor type option for a **ThermistorInputPoint**.

nrio-TriggerRateType

This topic contains a short description of a component or the component plugin view.

Contains rate calculation parameters for the parent NrioCounterInputProxyExt.

Index