Technical Document

Niagara$^{AX\text{-}3.4}$ NRIO Guide

May 5, 2009

Niagara^{AX} Nrio Guide

3951 Westerre Pkwy, Suite 350

Richmond

Virginia

23233

U.S.A.

Copyright Notice

Trademark Notices

Contents

Preface

Preface

NRIO FAQs

The following are frequently asked questions (FAQs) about the NiagaraAX NRIO driver:

Q: What does NRIO (or Nrio) abbreviate?

A: Niagara Remote Input Output, similar to the equivalent NDIO (or Ndio) "Niagara Direct Input Output" driver that is the station interface to other types of I/O hardware. Note throughout this document, capitalization of NRIO is typically limited to the leading "N", as in "Nrio proxy point". This reflects actual Niagara component and view names, such as NrioNetwork, Nrio16Module, Nrio Device Manager, and so on. Occasionally, the lowercase "nrio" is used when referring to the actual nrio module (as in .jar file).

Q: What is the difference between the Nrio and Ndio drivers?

A: To start with, each driver supports a different group of I/O hardware. Initially, the Nrio driver is needed only for the "onboard I/O" of a JACE-x02 Express or "M2M JACE". It will also be used for "remote I/O modules" as they are released. Both drivers use a "low-level daemon" to communicate to I/O processors on I/O hardware. Unlike the I^2C bus used by the Ndio daemon, the daemon in Nrio uses RS-485, and thus multiple NrioNetworks (each with a separate daemon) are possible on a JACE station—each NrioNetwork requires a specific COM port. Whereas, in Ndio, only *one* NdioNetwork is supported.

Also, in an NdioNetwork, all "devices" (NdioBoards) are automatically discovered and addressed relative to the physical *order of attachment* of I/O modules to the parent JACE. However, when RS-485 connected remote I/O devices (NrioModules) are discovered under an NrioNetwork, this proximity order does not exist. Instead, the driver automatically assigns consecutive addresses, and provides a "wink" action on each device so you can positively associate the discovered component to the actual physical I/O module.

Apart from these "network and device" differences, at the proxy point level there is little difference between Nrio and Ndio. Nrio and Ndio proxy points that represent hardware I/O points are nearly identical. There are fewer components in the Nrio palette, but the manager views in the Nrio driver allow you to create whatever new components are needed—even if working with the Nrio hardware offline.

Q: Why is it when I add some Nrio proxy points, say a VoltageInputPoint, that they come up in fault?

A: In the Nrio Point Manager if you add a VoltageInputPoint, ResistiveInputPoint, or ThermistorInputPoint and change the point's Facets to another category of units, you typically see this fault status. This happens because those types of proxy points are automatically added with a "LinearCalibrationExt", and its "units" property does not automatically stay in sync with the proxy point's Facets.

To clear this fault status, go to the property sheet of the LinearCalibrationExt under the point, and assign the identical units as you did in the proxy point's Facets. Alternatively, you could simply delete the point extension, but that would remove the "Offset" (calibration) utility that it provides.

Note this fault status happens most for VoltageInputPoints, and rarely (if ever) for a ThermistorInput-Point—because its units should always remain temperature, even if changing between °C and °F.

Document Change Log

Updates (changes/additions) to this *NiagaraAX NRIO Guide* document are listed below.

- Published: May 5, 2009
  Initial document.

Chapter 1

nrio Driver Installation

To use the NiagaraAX nrio driver, you must have a target JACE host that provides either onboard I/O hardware points using Nrio, such as an "M2M JACE" (JACE-x02 Express), and/or that supports external RS-485 connected I/O modules (when available). In addition, the "`nrio`" feature must be in the JACE's license.

From your PC, use the Niagara Workbench 3.*n.nn* installed with the "installation tool" option (checkbox "This instance of Workbench will be used as an installation tool"). This option installs the needed distribution files (*.dist* files) for commissioning various models of remote JACE platforms. The dist files are located under your Niagara install folder in a "sw\dist" subfolder.

Included in the dist file for each JACE platform is the necessary Nrio processor software, meaning underlying files and firmware used by a host JACE to communicate with either onboard (or RS-485 connected) Nrio modules. Nrio support is automatically installed when you use the "Distribution File Installer" in a platform session with the JACE, or the distribution file option in the platform's "Commissioning Wizard." Note that the dist file includes many *other* things besides just Nrio support.

Apart from installing the 3.*n.nn* version of the Niagara distribution file in the JACE, make sure to install the **`nrio`** and **`kitIo`** modules too (if not already present, or upgrade if an older revision). For more details, see "About the Commissioning Wizard" in the *JACE NiagaraAX Install and Startup Guide*.

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.*

Following this, the remote JACE is now ready for Nrio software configuration in its running station, as described in the rest of this document. See "About Nrio Architecture" on page 3-1.

Note: *Basic procedures for using the Nrio driver, including online discovery of I/O points (and their addition to your Niagara station), is in the next section. See "Nrio Quick Start" on page 2-1.*

Chapter 2

Nrio Quick Start

This section provides procedures to use the Nrio driver to make Nrio proxy points, the station interface to physical I/O points. As with other NiagaraAX drivers, you can do most configuration from special "manager" views and property sheets using Workbench.

These are the main subsections:

- Configure the Nrio network
- Create Nrio proxy points

Configure the Nrio network

Note: *At the time of this document, only one of the two types of Nrio networks is needed, for a specific purpose: an* **`M2mIoNetwork`** *to support the "onboard" I/O of an "M2M JACE" (JACE-202 Express).*

At some future date when remote RS-485 I/O hardware modules are released. you may need to add more than one Nrio network, using the other type of Nrio network: **`NrioNetwork`**.

To add and configure an Nrio network, perform the following main tasks:

- Add an M2mIoNetwork ("M2M JACE")
- Add and configure a NrioNetwork (future use)

Add an M2mIoNetwork ("M2M JACE")

To add an 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 in your Workbench palette side bar (see "Using the palette side bar" in the *User Guide* for general details).

Step 4 From the **`nrio`** palette, *drag* (or copy and paste) an **`M2mIoNetwork`** into the station's **`Drivers`** folder. In the popup **`Name`** dialog, 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.

If the station is running, within several seconds the previous copy triggers a discover, 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 When the are now ready to create Nrio proxy points. See "Create Nrio proxy points" on page 2-3.

For related details, see "About the M2mIoNetwork" on page 3-2.

Add and configure a NrioNetwork

Note: *This section is for "future use", when remote RS-485 I/O hardware modules become available. In this case, a different NrioNetwork is required for each RS-485 trunk with remote I/O modules.*

Each Nrio network must have a unique, specific Port Name *property* `COMn` *value. Also, each Nrio network must have a unique* Trunk *property value, starting at* `1`*, then* `2`*, and so on.*

For example:

- If 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 (onboard I/O): Port Name=COM3, Trunk=1 (both are fixed, read-only values)
  - NrioNetwork (onboard RS-485 wired to remote I/O modules): Port Name=COM2, Trunk=2

For further details, see "Essential Nrio network configuration properties" on page 3-3.

To add 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. For more details, see "Driver Manager New and Edit" in the *Drivers Guide*.

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. **`Save`** when done.

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

If necessary, add, configure, and save any additional NrioNetwork needed in a similar fashion—see the previous Note: for more details.

Step 6 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.

Discover and add NrioModules

To discover and add NrioModules

In Nrio architecture, NrioModules act as "device-level" components. An NrioModule represents an I/O *processor*, servicing some number of I/O points (see "About the NrioModule" on page 3-9).

Depending on JACE platform and installed devices, the number of NrioModules 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) wired to a specific RS-485 bus should discover the same number of NrioModules as I/O devices on that bus.

Note: *Currently, only one type of NrioModule is used:* **`Nrio16Module`***. This same component is used to model the 16 onboard I/O points of an M2M JACE as well as the 16 I/O points on a remote RS-485 I/O module (10-16-485).*

Step 1 Double-click the **`NrioNetwork`**, or:

right-click the NrioNetwork and select **`Views > Nrio Device Manager`**.

This brings up the **`Nrio Device Manager`**.

Step 2 Click the **`Discover`** button to launch an Nrio Board 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 (Figure 3-6 on page 5). The bottom pane, labeled "Database," is a table of NrioModules that are currently mapped into the Niagara station—initially (unless offline programming occurred), this pane with 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 an 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. For details, see "About Nrio Wink" on page 3-7, and "About Nrio Device Match" on page 3-7.

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

The **Add** dialog appears (see Figure 3-7 on page 6), in which you can enter a Description for each one, or edit that later.

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

See the next section: "Create Nrio proxy points".

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 NrioModules. You can then do a point discover under each NrioModule, and add points offline. For more details, see "About Nrio offline engineering" on page 3-6.*

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.

Note: *For general information, see "About the Point Manager" in the* Drivers Guide.

Using Discover to add Nrio proxy points

Use the following procedures:

- To discover I/O points
- To add Nrio point folders (optional)
- To add discovered I/O points as Nrio proxy points

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

To discover I/O points

Perform this task to discover I/O points.

Step 1 In the **Nrio Device Manager**, in the **Exts** column, double-click the Points icon 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.

Step 3 To add Nrio point folders and proxy points, see "To add Nrio point folders (optional)" and "To add discovered I/O points as Nrio proxy points".

To add Nrio point folders (optional)

Note: *Use of point folders to organize proxy points can provide benefits. Note point folders can be added at any time, and proxy points "moved" into them, if needed. However, if you add point folders before adding points, this can be a time saver.*

Perform this optional task to add Nrio point folders to contain and organize Nrio proxy points.

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**. (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 hierarchy.

Note: *Changes were made in the Nrio Point Manager that 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. For more information, see "Point folder / discovery notes" on page 3-12.*

To add 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.*

Perform this task to add Nrio proxy 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 You can map selected points in the station in different ways:

- Drag from the Discovered pane to Database pane (brings up an **Add** dialog).
- Double-click an item in the Discovered pane (also brings up an **Add** dialog).

This works the same as in other driver's Point Manager views, except for the optional target/selection of an Nrio point folder (typically, you need to work in the point manager view for that point folder).

Step 4 When the **Add** dialog appears, you typically edit a number of fields for each I/O point. For complete details, see "Add and Edit dialog fields" on page 3-13.

The following brief summaries explain Add dialog fields:

- **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. See "Universal Input selection notes" on page 3-13 and "Output type selection notes" on page 3-14.
  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. See "Conversion types in Nrio proxy points" on page 3-15.
- **Facets** are the Nrio proxy point's facets, for how the value should be displayed in Niagara.

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

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

For more details, see "About Nrio proxy points" on page 3-14.

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.*

For more information, see "Linear Calibration Ext" on page 3-19.

Chapter 3

Niagara Nrio Concepts

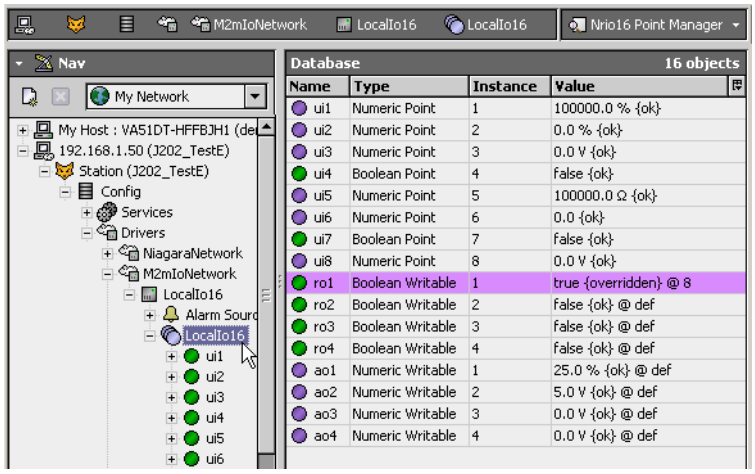
This section provides conceptual details on the Nrio driver (Niagara remote input/output) and its components, including views. Nrio components are the station interface to certain JACE controllers with onboard I/O points, such as the "M2M JACE" (JACE-202 Express), and/or remote I/O expansion modules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

About Nrio Architecture

Essentially, Nrio uses the standard NiagaraAX network architecture. See "About Network architecture" in the *Drivers Guide* for more details. 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 3-1).

Figure 3-1 *Nrio driver architecture*



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.*

For more details see:

- About the Nrio network
- Nrio Device Manager
- About the NrioModule
- Nrio Point Manager
- About Nrio proxy points

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 is used: the M2mIoNetwork, which supports the integral "onboard I/O" on an "M2M JACE" (JACE-202 Express). Later, as remote I/O modules (10-16-485) become available, there may be multiple Nrio access paths, and thus a need for 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.*

Any Nrio network should reside in the station's DriverContainer ("Drivers"). 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 to add an M2mIoNetwork—see "About the M2mIoNetwork" for details.

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. For details, See "Common network components" in the *Drivers Guide*. The following sections explain further:

- About the M2mIoNetwork
- Essential Nrio network configuration properties
- Nrio Network status notes
- Nrio Network monitor notes
- Nrio network tuning policy notes
- Nrio network poll scheduler notes
- Nrio network views
- About the actrld (access control daemon)

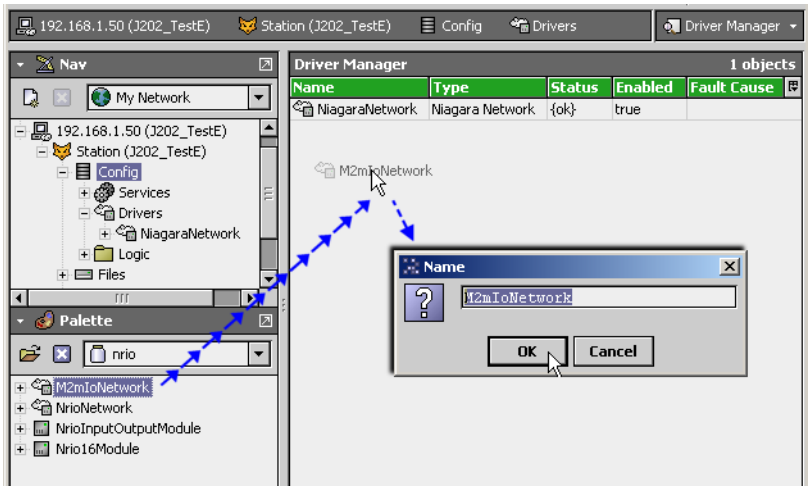
About the M2mIoNetwork

The M2mIoNetwork is a specialized version of the NrioNetwork, pre-configured to communicate to the onboard I/O processor of an "M2M JACE" (JACE-x02 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 DriverContainer ("Drivers"). The recommended way is to simply copy (drag and drop) the M2mIoNetwork from the `nrio` palette into Drivers. See Figure 3-2.

Figure 3-2 *Dragging M2mIoNetwork from nrio palette into station's DriverContainer*



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. For related details, see "NrioModule properties" on page 3-9.

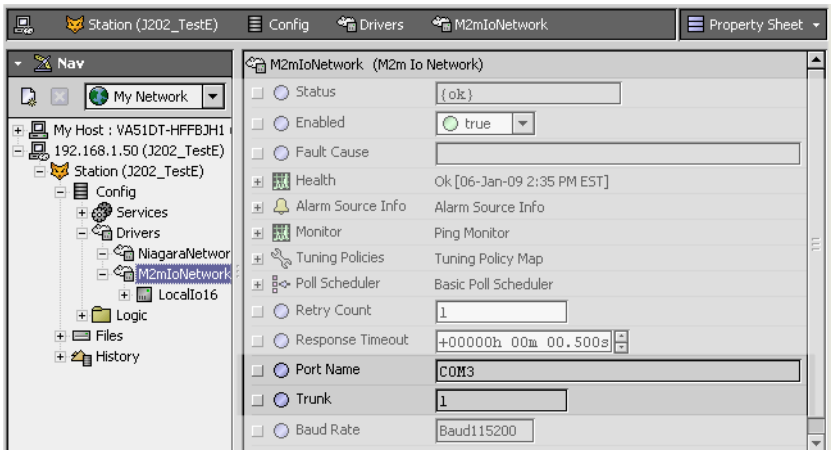
The M2mIoNetwork and NrioNetwork components have the typical collection of slots and properties as most other network components. For details, See "Common network components" in the *Drivers Guide*.

The following sections explain further:

Essential Nrio network configuration properties

There are two important configuration properties for any Nrio network; access them on the network's property sheet view, as shown in Figure 3-3.

Figure 3-3 *Essential Nrio network properties Port Name, Trunk*



- **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 3-1.

Table 3-1 *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
"M2M JACE" (JACE-x02 Express)	COM3	COM2	COM7, COM8
JACE-7 series	—	COM2	COM3, COM4

- **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 "actrld*n*" (access control daemon) that polls that IO. For related details, see "About the actrld (access control daemon)" on page 3-4.

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=COM3 and Trunk=1.*

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).

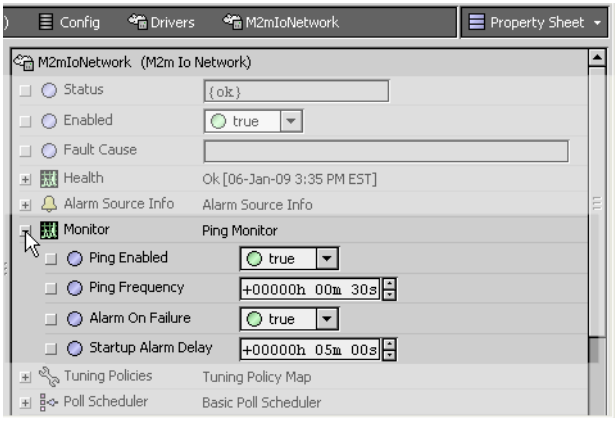
The Health slot contains historical timestamp properties that record the last network status transitions from ok to any other status. The "Fault Cause" property further explains any fault status.

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. See "About network Alarm Source Info" in the* Drivers Guide *for more details.*

Nrio Network monitor notes

The network's monitor routine verifies child NrioModule component(s)—the "pingable" device equivalent in Nrio. For general information, see "About Monitor" in the *Drivers Guide*.

Figure 3-4 *Monitor properties of an Nrio network*



Nrio network tuning policy notes

An Nrio network has the typical network-level Tuning Policy Map slot with a single default Tuning Policy, as described in "About Tuning Policies" in the *Drivers Guide*.

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, as described in "About poll components" in the *Drivers Guide*. 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. See "About the actrld (access control daemon)" for related details.

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 NrioModule components to the station. For details, see "Nrio Device Manager" on page 3-5.

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. See "About IO processor updates to IoStatus" for related details.

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.

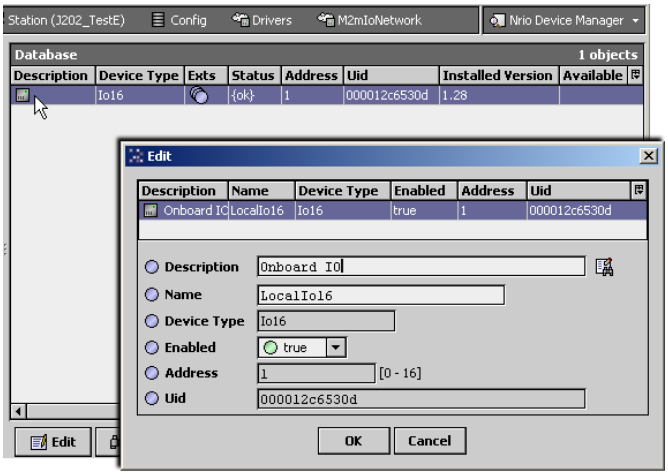
About IO processor updates to IoStatus

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 3-5 *M2mIoNetwork in Nrio Device Manager*



As shown in Figure 3-5, 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 the **`Exts`** icon 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 (Figure 3-6). In this case, this view allows you to discover and add one or more "device-level" NrioModule components to the station database (Figure 3-7).

Figure 3-6 *Adding NrioModule discovered using Nrio Device Manager*

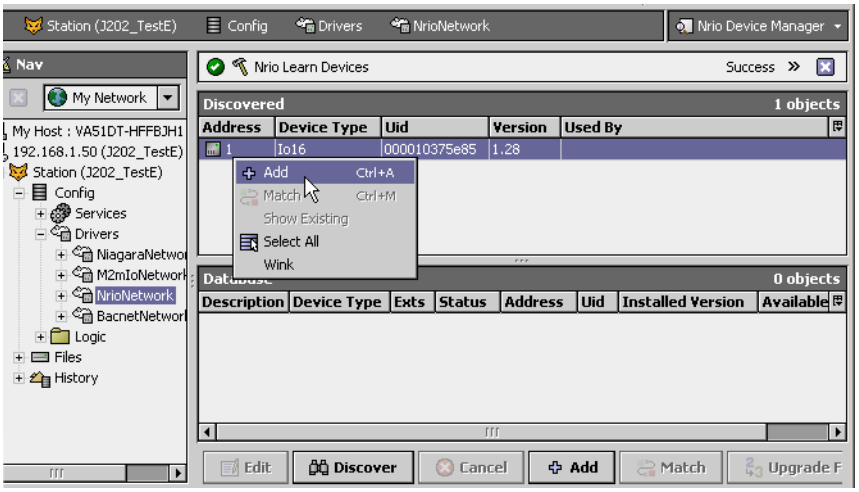
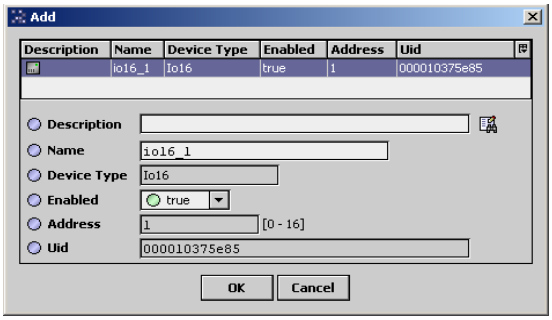


Figure 3-7 *Add dialog for NrioModule(s)*



For general information, refer to "About the Device Manager" in the *Drivers Guide*.

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).

See the following sections for more details on the Nrio Board Manager view:

- Nrio Device Manager usage notes
- About Nrio offline engineering
- About Upgrade Firmware

Nrio Device Manager usage notes

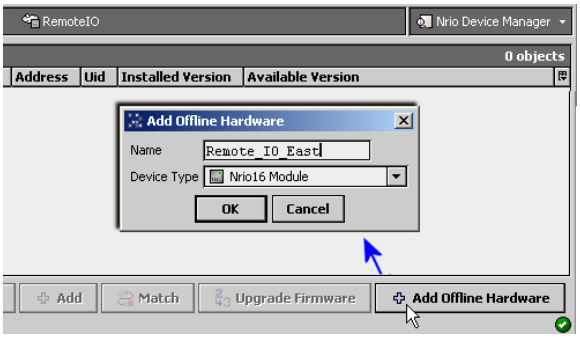
Note the following when using the Nrio Device Manager:

- For an M2mIoNetwork, only *one* child Nrio16Module is valid. This supports the onboard IO of an "M2M JACE" (JACE-x02 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 (Figure 3-7) 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). For details, see "About the NrioModule" on page 3-9.
- 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. For more details, see "About Nrio offline engineering", including "About Nrio Wink" and "About Nrio Device Match".

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. Figure 3-8 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 3-8 *Add Offline Hardware in Nrio Device Manager*



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).

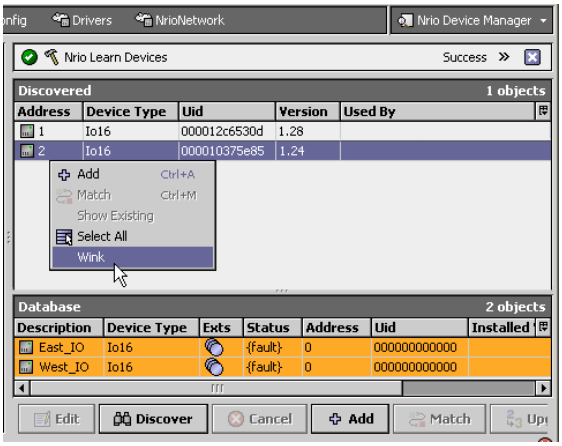
Note: *Following adding an NrioModule using the* **Add Offline Hardware** *method, note you can still use "Learn" mode from the component's Points extension's Nrio Point Manager view to "Discover" and Add Nrio proxy points (all will have a* fault *status). And as needed, add other extension types (e.g. 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 NrioModules on the NrioNetwork, you typically "wink" discovered IO devices before making each match. For related details, see "About Nrio Wink" and "About Nrio Device Match".

About Nrio Wink

Discovered Nrio devices have an available right-click Wink action, as shown in Figure 3-9.

Figure 3-9 *Wink function is used to verify a discovered device*



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.

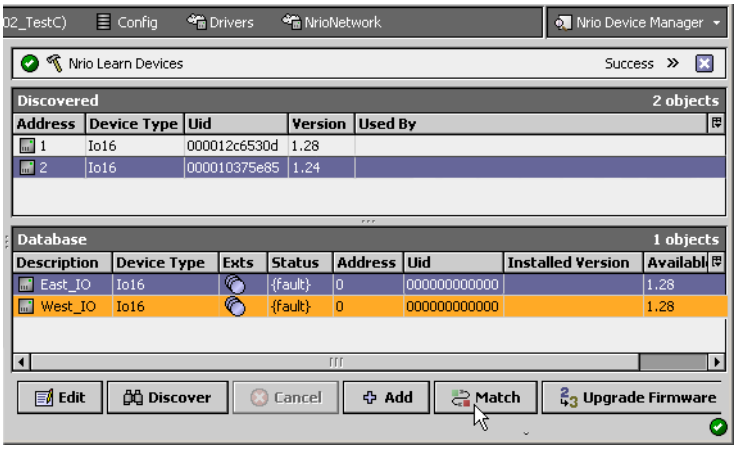
Following a device Wink to confirm the location of a device, typically you perform a match. See "About Nrio Device Match".

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. For general information, refer to the "Match (Device)" section in the *Drivers Guide*.

Figure 3-10 shows a match about to be initiated between a discovered Nrio16Module and an offline-created Nrio16Module.

Figure 3-10 *Match function is used to associate discovered device with offline-created NrioModule*



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).

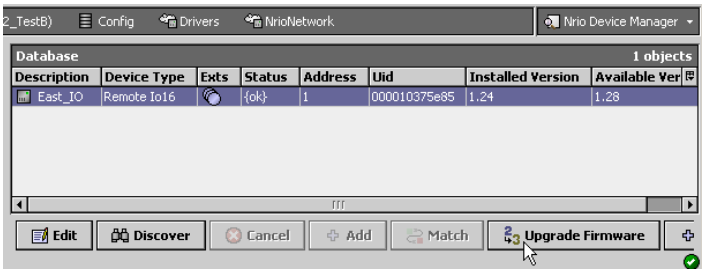
About Upgrade Firmware

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.

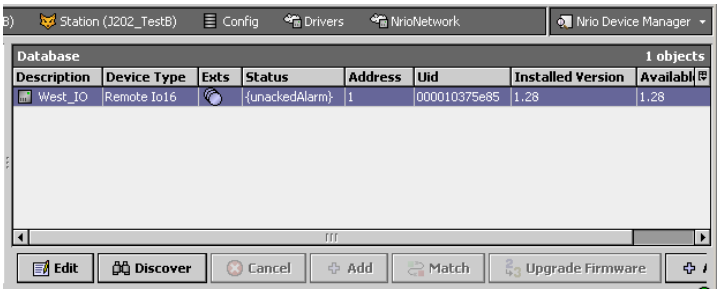
Figure 3-11 *Upgrade Firmware in Nrio Device Manager*



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 3-12).

Figure 3-12 *Following a firmware upgrade in Nrio Device Manager*

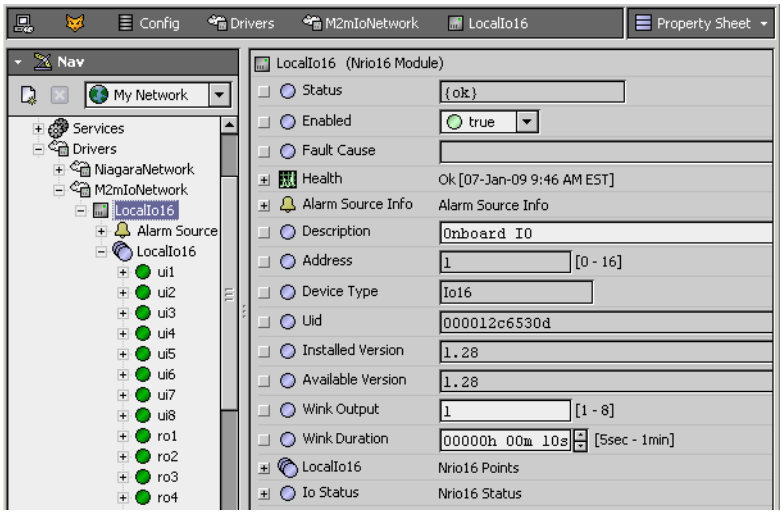


Note also that device "Ping" alarms typically result from an IO firmware upgrade—from when the NrioModule(s) toggled offline and back online.

About the NrioModule

The NrioModule represents 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 3-13 *NrioModule (Nrio16Module) property sheet*



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). If a JACE has 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 NrioModule must have a unique "Address" property value, from 1 to 16. This address value is automatically derived during an online Discover, and is *read-only*. Also during a 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, only the wink-related properties are writable (Wink Output, Wink Duration). All other NrioModule properties are read-only types (Figure 3-13).

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. For a procedure, see "Configure the Nrio network" on page 2-1.*

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). For more details, see "Nrio Point Manager" on page 3-11.

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. See "Device Alarm Source Info" in the* Drivers Guide *for more details.*

Device status properties

An NrioModule has typical device-level status properties (see "Device status properties" in the *Drivers Guide*). The following notes apply:

- **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 above.
- **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

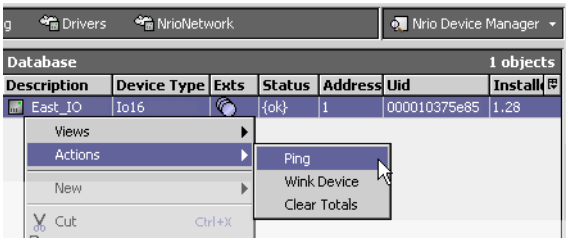
In addition to common device-level Device status properties, NrioModule has the following unique properties, where most are read-only types unless noted:

- **Address**
  Integer between 1 and 16, unique among all NrioModules under the NrioNetwork. Automatically derived (and associated) with physical I/O devices upon an online Discover.
- **Device Type**
  Currently, this is `Io16`. Other device types may be available as other I/O hardware products (accessed using Nrio) are offered.
- **Uid**
  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`**. For more details, see "About Upgrade Firmware" on page 3-8.
- **Wink Output**
  (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).
- **Wink Duration**
  (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. For related details, see "About Nrio Wink" on page 3-7.*
- 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. For more details, see the section "Nrio Point Manager" on page 3-11.
- **Io Status**
  (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. For related details, see "About the actrld (access control daemon)" on page 3-4.*

NrioModule actions

Each NrioModule has three right-click actions, as shown for the Nrio16Module in Figure 3-14.

Figure 3-14 *Actions for an Nrio16Module*



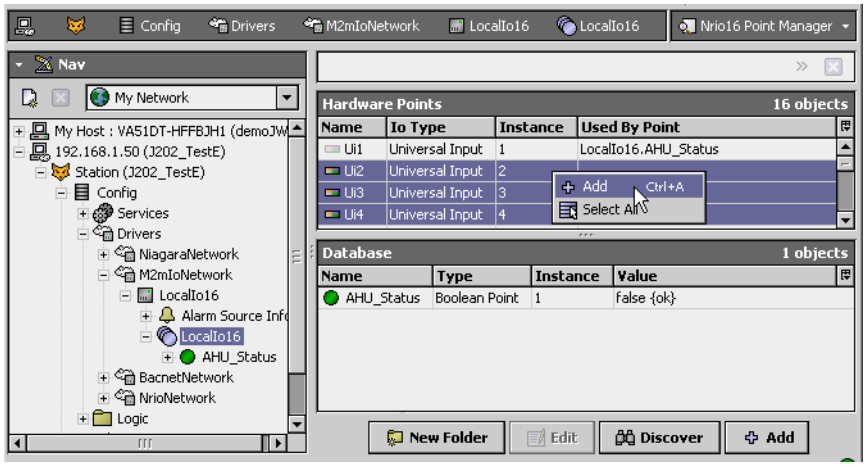
These actions are summarized as follows:

- **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. For related details, see "About Nrio Wink" on page 3-7.
- **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. For related details, see "CounterInputPoint" on page 3-21.

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 (Figure 3-15). You use this view to discover and add corresponding Nrio proxy points to the station database.

Figure 3-15 *Adding Nrio proxy points discovered using Nrio16 Point Manager*



Typically you add a proxy point for *each* discovered I/O terminal. For more details, see "Nrio Point Manager usage notes" on page 3-11. For procedures, see "Create Nrio proxy points" on page 2-3.

For general information about point managers, see "About the Point Manager" in the *Drivers Guide*.

Nrio Point Manager usage notes

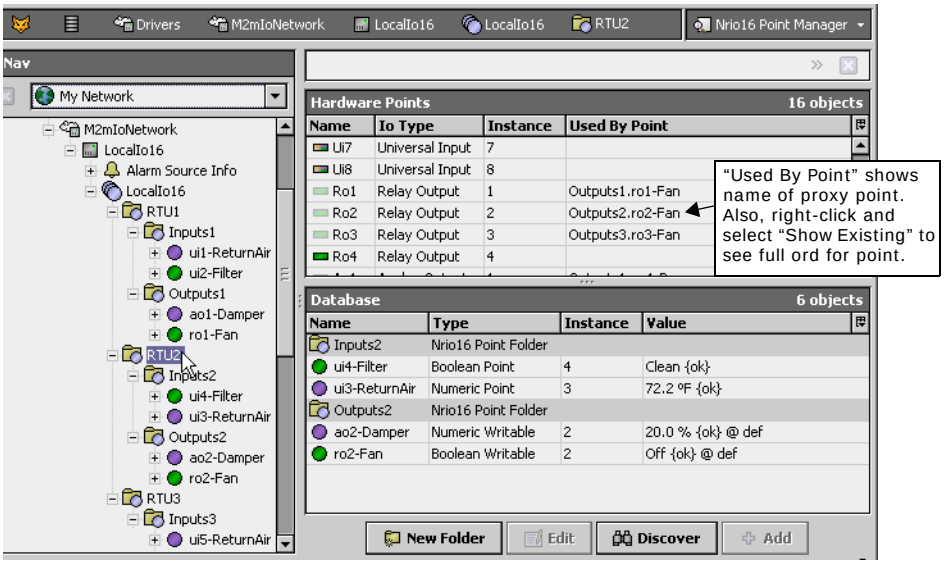
The following notes apply when using the Nrio Point Manager:

- Point folder / discovery notes
- Add and Edit dialog fields
- Universal Input selection notes
- Output type selection notes

Point folder / discovery notes

Like many NiagaraAX 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. Figure 3-16 shows such an Nrio Point Manager view.

Figure 3-16 *Nrio16 Point Manager view for an Nrio Point Folder*



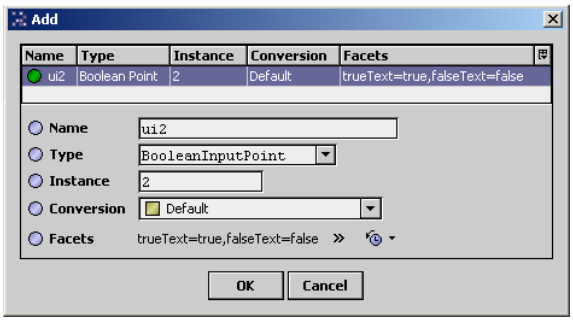
However, a few *changes* are unique to Nrio Point Manager views, summarized below.

- **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 point 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 (Figure 3-17), and afterwards, in the **Edit** dialog.

Figure 3-17 *Add dialog in Nrio Point Manager*



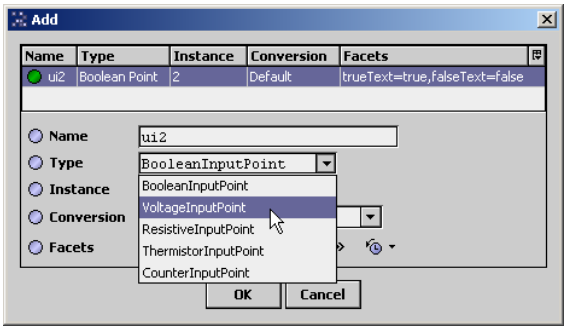
- **Name**
  Name of the Nrio proxy point (equivalent to right-click Rename, can be edited anytime). The default name appears as "*AbbrevIoTypeTerminalNumber*", for example, "ui4", "ao2", and so on. Typically, you edit name from the default name to reflect the actual purpose of the I/O point, such as "Room101T," "AHU1_FanStatus," and so forth.
- **Type**
  The type of Nrio proxy point to create (*not* editable after adding, see "Universal Input selection notes" on page 3-13.)
- **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. See "Conversion types in Nrio proxy points" on page 3-15 for more details.
- **Facets**
  Facets for the Nrio proxy point. Equivalent to accessing facets through the point's property sheet (and can be edited anytime). For more details, see "About point facets" in the *Drivers Guide*.

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 Figure 3-18.

Figure 3-18 *Select Nrio input type for each universal input*



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. For more details, see "About Nrio proxy points" on page 3-14.

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, as shown in Figure 3-18. 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. For general information, see "About proxy points" in the *Drivers Guide*.

The following sections explain further:

- Nrio point types
- Nrio Proxy Ext common properties
- Conversion types in Nrio proxy points
- Scale and offset calculation (linear)
- Non-linear sensor support
- Linear Calibration Ext

Nrio point types

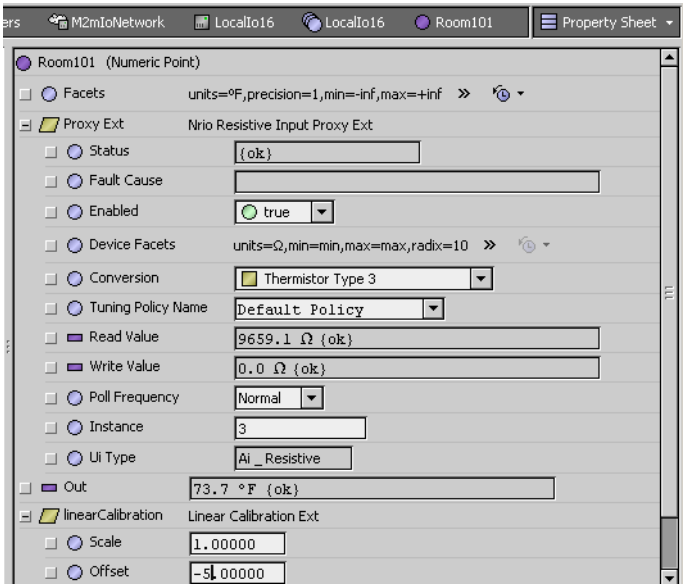
There are 7 different Nrio point types, which derive from the following 4 control point types:

- **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 3-19 *Common properties among Nrio proxy extensions*



As shown in the property sheet view (Figure 3-19), 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). For details see "Conversion types in Nrio proxy points" on page 3-15.
- **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, with the following available selections:

- 500 Ohm Shunt (for 4-20mA sensor using VoltageInputPoint)
- Default
- Generic Tabular (requires `kitIo` module to be installed)
- Linear
- Reverse Polarity

- Tabular Thermistor
- Thermistor Type 3

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.*

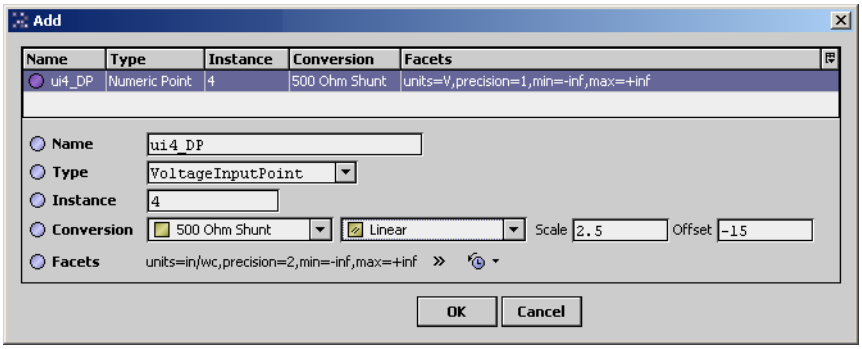
500 Ohm Shunt

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. Figure 3-20 shows an example configuration of such a point.

Figure 3-20 *Example VoltageInputPoint used for 4-to-20mA sensor (Edit dialog from Point Manager)*



  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.
  For related details, see "Non-linear sensor support" on page 3-18.

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. See "Scale and offset calculation (linear)" on page 3-17 and "Linear Calibration Ext" on page 3-19.

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 table" method 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. For more details, see "Non-linear sensor support" on page 3-18 and "Curve File Import/Export notes" on page 3-25.

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)" on page 3-17.

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. See "500 Ohm Shunt" on page 3-16.*

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 trouble-shooting 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. For details, see "Tabular Thermistor notes" on page 3-24 and "Curve File Import/Export notes" on page 3-25.

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`

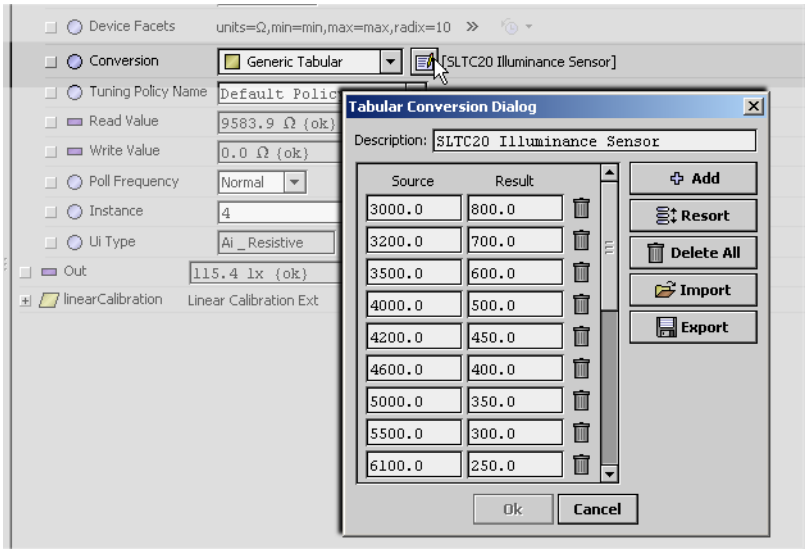
Figure 3-20 shows this point's configuration in a Workbench dialog. Note that the Linear Calibration Ext for this point will require its "Units" property to be configured to match the facets for this point—otherwise, the point will have a fault status. See the "Linear Calibration Ext" section for related 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 Figure 3-21.

Figure 3-21 *Generic Tabular Dialog from edit control*



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. Figure 3-21 shows a point with a non-linear curve described in the "Non-linear sensor example" on page 3-19.

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 sta-

tion (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.*

Related to this, see "Curve File Import/Export notes" on page 3-25.

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`** (see Figure 3-21 on page 18).

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.

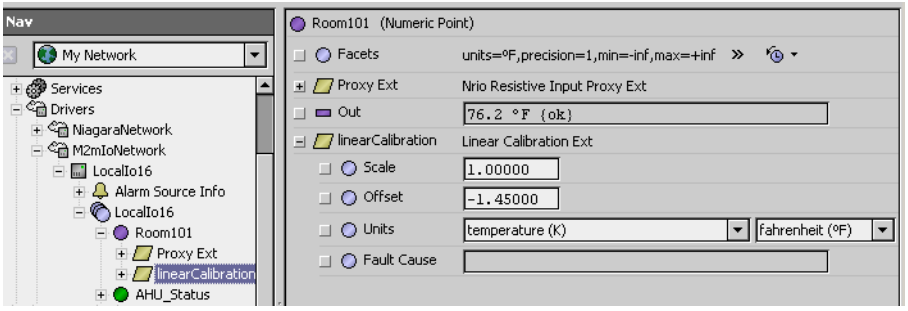
Table 3-2 *Example illuminance sensor source (ohms) to results (lux)*

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 Figure 3-22.

Figure 3-22 *Linear Calibration Extension*



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.`

Note: *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:*

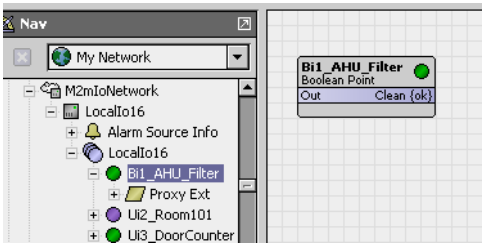
- Facets: gallons [units: volume (m^3), gallon (gal)]
- NrioCounterInputProxyExt:
  - Conversion: Default
  - Output Select: Count
  - Rate Calc Type: FixedWindowRateType
  - Rate Calc:
    - Scale: 9.0
    - Interval: 15s
- LinearCalibrationExt:
  - Scale: 0.15
  - Offset: 0.0

*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 3-23 *Nrio Boolean Input Point*



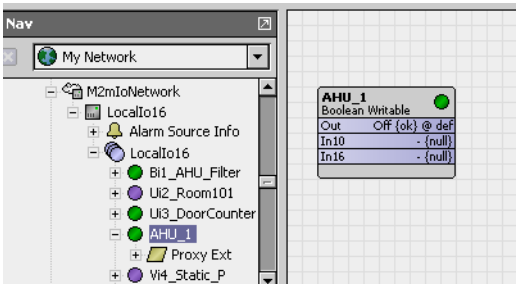
The BooleanInputPoint contains only common Nrio Proxy Ext properties. Among these, note that the "Device Facets" property is blank by default.

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 (for details, see "About writable points" in the *User Guide*).

Figure 3-24 *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.

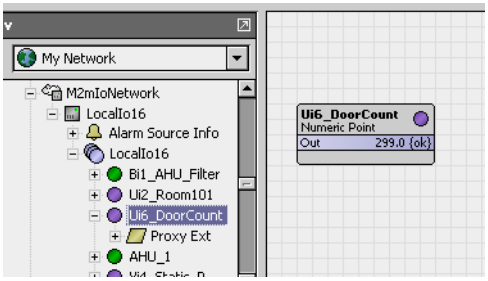
CounterInputPoint

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 3-25 *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:

- FixedWindowRateType
- SlidingWindowRateType
- TriggerTypeRate

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" on page 3-19, 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" on page 3-21):
  - **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 unscaled (actual number of pulses) value, so if a scaled Count is configured, it will show differently.

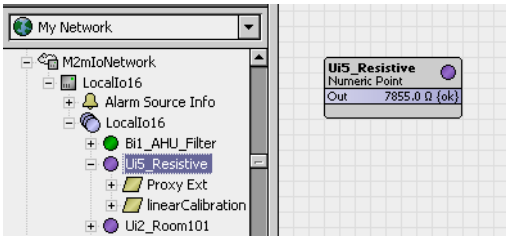
ResistiveInputPoint

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. For details, see "Scale and offset calculation (linear)" on page 3-17 and "Non-linear sensor support" on page 3-18.

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. For details, see "Linear Calibration Ext" on page 3-19.

Figure 3-26 *Nrio Resistive Input Point*



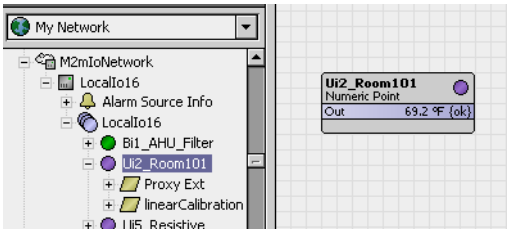
The ResistiveInputPoint's Proxy Ext contains only common Nrio Proxy Ext properties. Among these, note that the read-only "Device Facets" property is preset to ohms.

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. For details, see "Linear Calibration Ext" on page 3-19.

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 3-27 *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":

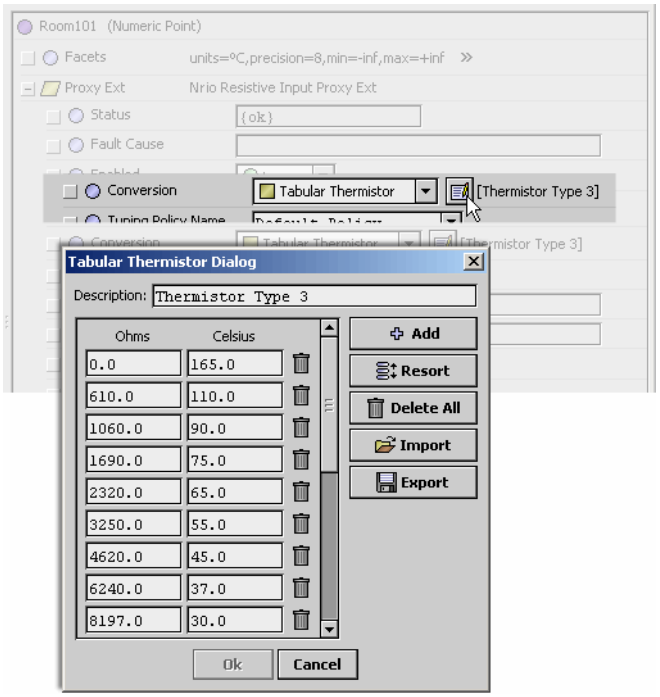
Table 3-3 *Type-3 Thermistor Nrio Curve*

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
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 in Figure 3-28.

Figure 3-28 *Tabular Thermistor Dialog from edit control*



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.*

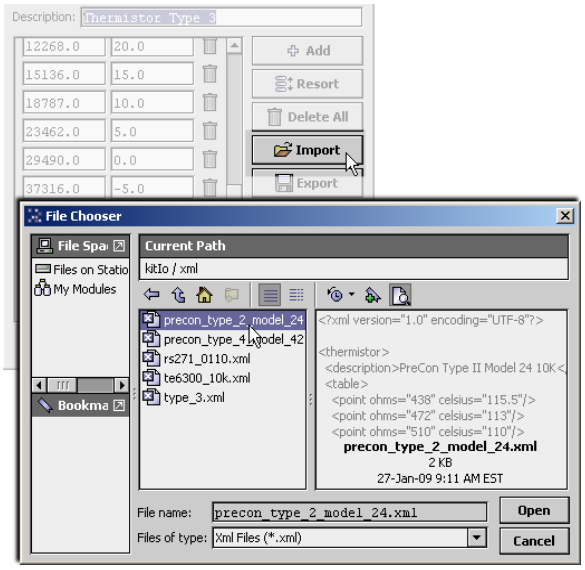
  Related to this, see Curve File Import/Export notes.

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** (Figure 3-28) for a "Tabular Thermistor" conversion, or **Generic Tabular Dialog** (Figure 3-28 on page 24) for a "Generic Tabular" conversion.

When you do this, the standard Niagara **File Chooser** dialog appears, as shown in Figure 3-29.

Figure 3-29 *File Chooser to locate thermistor .xml file for Import*



To *import* a thermistor curve file, navigate to the `xml` folder in the **kitIo** module (Jar) file in your modules ("My Modules"), see Figure 3-29. 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—see "Non-linear sensor support" on page 3-18.

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.*

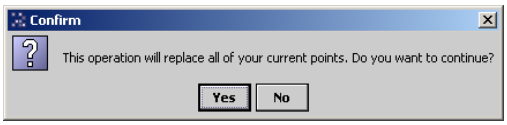
As of the date of this document, the **kitIo** module contains an `xml` folder with five (5) different thermistor curves in xml format, as follows:

- `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

Note: *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, as shown in Figure 3-30*

Figure 3-30 *Curve file import confirmation*



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.

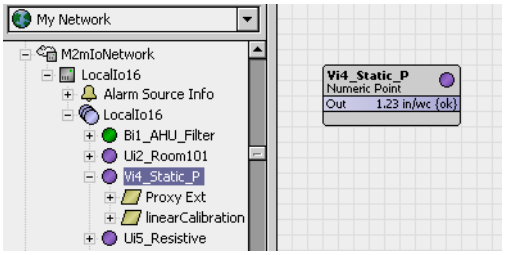
VoltageInputPoint

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. See "Scale and offset calculation (linear)" on page 3-17.

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. See "Conversion types in Nrio proxy points" on page 3-15 for more details.*

The VoltageInputPoint is pre-configured with a linearCalibration extension, to allow for offset correction. For details, see "Linear Calibration Ext" on page 3-19.

Figure 3-31 *Nrio Voltage Input Point*

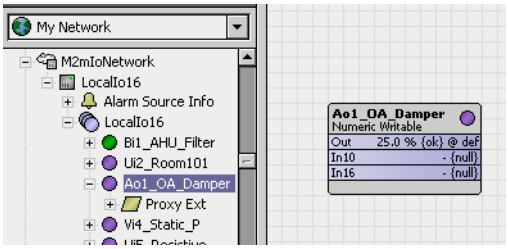


The VoltageInputPoint's Proxy Ext contains only common Nrio Proxy Ext properties. Among these, note that the read-only "Device Facets" property is preset to volts (V).

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 (see "About writable points" in the *User Guide*).

Figure 3-32 *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)" on page 3-17.

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

Note: *The following four Hardware Override properties were intended for a I/O hardware module that was never released with NiagaraAX support. 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).
  Note: *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 4

Nrio Plugin Guides

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 Plugin Guides Summary

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

- NrioDeviceManager
- Nrio16PointManager

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. For more details, see "Nrio Device Manager" on page 3-5.

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.
  Note: *An M2mIoNetwork supports only a single Nrio16Module—it always has 1 for Address.*
- **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.
- **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 NrioPointFolder 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. For more details, see "Nrio Point Manager" on page 3-11.

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 NiagaraAX 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.

Chapter 5

Nrio Component Guides

These component guides provides summary help on Ndio components.

Nrio Component Reference Summary

Summary information is provided on components in the `nrio` module, listed alphabetically as follows:

- FixedWindowRateType
- LinearCalibrationExt
- M2mIoNetwork
- Nrio16Module
- Nrio16Points
- NrioBooleanInputProxyExt
- NrioCounterInputProxyExt
- NrioNetwork
- Nrio16PointFolder
- NrioPollScheduler
- NrioRelayOutputProxyExt
- NrioResistiveInputProxyExt
- NrioVoltageOutputProxyExt
- NrioVoltageInputProxyExt
- SlidingWindowRateType
- ThermistorType3Type
- TriggerRateType

nrio-FixedWindowRateType

Contains rate calculation parameters for the parent NrioCounterInputProxyExt. For details, see "FixedWindowRateType" on page 3-21 and "About Nrio rate calculation" on page 3-21.

nrio-LinearCalibrationExt

LinearCalibrationExt is an point extension that allows for calibration adjustment of an input to a known measured value. For more details, see "Linear Calibration Ext" on page 3-19.

nrio-M2mIoNetwork

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. For details, see "About the M2mIoNetwork" on page 3-2.

nrio-Nrio16Module

Nrio16Module represents either the onboard I/O of a JACE or an external I/0 module (10-16-485), and is currently the only supported type of **`NrioModule`**. You use the Nrio16PointManager view of its *child* Nrio16Points extension to discover, add, and edit Nrio proxy points. For more details, see "About the NrioModule" on page 3-9.

nrio-Nrio16Points

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 Nrio16Points 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. For details, see "Nrio Point Manager" on page 3-11.

nrio-NrioBooleanInputProxyExt

NrioBooleanInputProxyExt is the proxy extension for a `BooleanInputPoint`. It represents an Nrio *boolean universal input*. For details, see "BooleanInputPoint" on page 3-20.

nrio-NrioCounterInputProxyExt

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. For more details, see "CounterInputPoint" on page 3-21.

nrio-NrioNetwork

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. For more details, see "About the Nrio network" on page 3-2.

nrio-Nrio16PointFolder

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. For related details, see "Point folder / discovery notes" on page 3-12.

nrio-NrioPollScheduler

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. For more details, see "Nrio network poll scheduler notes" on page 3-4.

nrio-NrioRelayOutputProxyExt

NrioRelayOutputProxyExt is the proxy extension for a `RelayOutputWritable`. It represents an Nrio *boolean output*. For details, see "RelayOutputWritable" on page 3-21.

nrio-NrioResistiveInputProxyExt

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. For more details, see "ResistiveInputPoint" on page 3-23 and "ThermistorInputPoint" on page 3-23.

nrio-NrioVoltageInputProxyExt

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). For details, see "VoltageInputPoint" on page 3-26.

nrio-NrioVoltageOutputProxyExt

NrioVoltageOutputProxyExt is the proxy extension for an `VoltageOutputWritable`. It represents an Nrio *analog output* (AO) that produces 0-to-10Vdc. For details, see "VoltageOutputWritable" on page 3-26.

nrio-SlidingWindowRateType

Contains rate calculation parameters for the parent NrioCounterInputProxyExt. For details, see "SlidingWindowRateType" on page 3-22 and "About Nrio rate calculation" on page 3-21.

nrio-ThermistorType3Type

ThermistorType3Type is a thermistor type option for a **`ThermistorInputPoint`**. For more details, see "ThermistorInputPoint" on page 3-23.

nrio-TriggerRateType

Contains rate calculation parameters for the parent NrioCounterInputProxyExt. For details, see "TriggerTypeRate" on page 3-22 and "About Nrio rate calculation" on page 3-21.