Technical Document

Niagara$^{AX-3.x}$ Lonworks Guide

June 7, 2013

Niagara^{AX} Lonworks Guide

Confidentiality Notice

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

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

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

Trademark Notice

BACnet and ASHRAE are registered trademarks of American Society of Heating, Refrigerating and Air-Conditioning Engineers. Microsoft and Windows are registered trademarks, and Windows NT, Windows 2000, Windows XP Professional, and Internet Explorer are trademarks of Microsoft Corporation. Java and other Java-based names are trademarks of Sun Microsystems Inc. and refer to Sun's family of Java-branded technologies. Mozilla and Firefox are trademarks of the Mozilla Foundation. Echelon, LON, LonMark, LonTalk, and LonWorks are registered trademarks of Echelon Corporation. Tridium, JACE, Niagara Framework, Niagara^{AX} Framework, and Sedona Framework are registered trademarks, and Workbench, WorkPlace^{AX}, and ^{AX}Supervisor, are trademarks of Tridium Inc. All other product names and services mentioned in this publication that is known to be trademarks, registered trademarks, or service marks are the property of their respective owners.

Copyright and Patent Notice

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

Copyright © 2010 Tridium, Inc.

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

Contents

PREFACE

Preface

Document Change Log

Updates (changes/additions) to this *Niagara*^{AX} *Lonworks Guide* document are listed below.

- Updated June 7, 2013
  Several security-related notes and cautions were added in the "Lon tunneling" appendix, including in subsections "Lon tunnel overview" on page B-1, "Lon tunnel client configuration" on page B-3, and "Configuring the Lon tunnel server" on page B-4. A new subsection "Best security practices for tunneling" on page B-5 provides related configuration details.
- Updated June 22, 2010
  Updates mostly reflecting AX-3.5-related changes. Removed many references to early NiagaraAX release or build levels (AX-3.0, AX-3.1, and AX-3.2), where AX-3.3 is considered baseline. New or changed areas of the document include a new section about Lon proxy point "Facet support for "null" output" on page 3-5, and changes in the "About the Lon Utilities Manager" on page 3-25 section that shows an updated figure for building a command, as well as a description for the Read Mem command. Changes were made in the section "Notes when configuring as Lon node" on page 3-40, in subsections "Add Local nvs and ncis", "Notes on editing LocalNvs and LocalNcis", and "Local Nv Manager notes". Changes include descriptions of additional properties in the New / Edit dialog for Local Nvs and Local Ncis, available starting in AX-3.5.
  The "Lon Component Guides" section had new or updated summary descriptions added for components "kitLon-BufferParams" on page 5-1, "lonIp-ChannelMember" on page 5-2, and "lonIp-Ip-LonNetworkConfig" on page 5-2.
  A new appendix "Lon over IP" on page C-1 was added, which contains more details about using the lonIp module, in addition to the brief summary descriptions in "Components in lonIp module". Starting in AX-3.5, the lonIp module was extended with the ability for the Niagara (JACE) station to act as the CEA-852 "Configuration Server". Main sections in the new appendix include "Overview", "LonIp Setup", "About the IpLonNetwork's Member Table", and "About ChannelMembers".
- Updated: April 18, 2008
  Updates reflecting AX-3.4-related changes, including a new Workbench "ExtractXif" command on a LocalLonDevice, a new kitLon component, and a new "zero-based" array indices option when creating an `.lnml` file. See "LocalLonDevice XIF generation (AX-3.4 and later)" on page 3-45 , "kitLon-LonReplace" on page 5-2, and the first note in "Lon Xml creation" on page A-2. A minor change was made in "Lon Comm Config notes" on page 3-6. Details were added about the LinkFilter (debug usage) component, see "lonworks-LinkFilterView" on page 4-1 and "lonworks-LinkFilter" on page 5-5.
- Updated: February 14, 2008
  Changed document to reference the new *NiagaraAX Drivers Guide*, which was created from information formerly found in the NiagaraAX *User Guide*.
- Updated: December 18, 2007
  Updates reflecting AX-3.3-related changes. In the concepts chapter, the "Lon Objects" feature was explained in a new section "LonObjects" on page 3-36, and this also affected existing sections as follows: "About Lon Netmgmt" on page 3-9, "ImportXml command" on page 3-30, "Learn Nv" on page 3-35, and "About the Nv Manager" on page 3-31.
  In the "Lon Component Guides", a brief description was added for the new LonPoint component found in the kitLon module.

Note: *Update change log entries prior to December 18, 2007 were removed. AX-3.3 or later is now considered "baseline" in most descriptions in this document.*

Chapter 1

Lonworks Driver Installation

To use the NiagaraAX Lonworks driver, you must have a target JACE host that is licensed with the "`lonworks`" feature. In addition, other lonworks device limits or proxy point limits may exist in your 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 directory under a "sw" subdirectory. For details, see "About your software database" in the *Platform Guide*.

Apart from installing the 3.*n.nn* version of the Niagara distribution in the JACE, make sure to also install the *lonworks* module, plus any specific lon<*vendor*> module needed (for example, lonHoneywell, lonSiebe, lonStaefa, etc). Note that a *kitLon* module is also available. Upgrade any modules shown as "out of date". For details, see "Software Manager" in the *Platform Guide*.

Note: *If using the Quik Learn feature in the Lon Device Manager, it is necessary to have any needed lon<vendor> module installed in the remote JACE. The Quik Learn routine is performed by the station running on that JACE, and so requires "local access" to lon<vendor> modules. Note this varies from a Discover and Add operation, which references modules in your Workbench (client PC's) software database.*

Note: *If you need "Lon tunneling," install the* `lonTunnel` *module in the JACE also. See "Lon tunneling" on page B-1 for more details.*

Note: *If using the LON over IP driver (LonIp) install the* `lonIP` *module in the JACE also. See "Lon tunneling" on page B-1 for more details.*

Following this, the remote JACE is now ready for LonNetwork configuration in its running station, as described in the rest of this document. See "Niagara Lonworks Concepts" on page 3-1.

Basic procedures for using the online features of the driver, including discovery of Lon nodes and use of Quik Learn, is in the next section. See "Lonworks Quick Start" on page 2-1.

Chapter 2

Lonworks Quick Start

This section provides a collection of procedures to use the NiagaraAX Lonworks driver in typical online scenarios. Like other NiagaraAX drivers, you can do most configuration from special "manager" views and property sheets using Workbench. These are the main subsections:

- Configure the LonNetwork
- Discover and learn Lon devices
- Create offline database and match
- Create Lon proxy points
- Bind Lon proxy points

Configure the LonNetwork

To configure the LonNetwork, perform the following main tasks:

- Add a LonNetwork
- Configure the working domain (only if needed)

Add a LonNetwork

To add a LonNetwork in the station

Use the following procedure to add a LonNetwork component under the station's Drivers container.

Note: *If the host JACE has multiple LON ports, add one LonNetwork for each physical port. Under each LonNetwork, in* **`Lon Comm Config`***, specify the port's Device Name (LON1, LON2).*

To add a LonNetwork in the station:

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 "`LonNetwork`," number to add: `1` and click **`OK`**.

This brings up a dialog to name the network.

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

You should have a LonNetwork named "LonNetwork" (or whatever you named it), under your Drivers folder, showing a status of "`{ok}`" and enabled as "`true`."

To perform online discovery, see "Discover and learn Lon devices" on page 2-2.

Configure the working domain

It may be necessary to configure the working domain before it is possible to communicate with devices. If you are unable to discover nodes that are physically connected to the station, they may be configured on a different domain. Use the following method to gain access to a device by use of its Lonworks service pin.

To identify a device using its service pin message

To identify a device using a service pin message, in the station with the LonNetwork:

Step 1 Right-click the LonNetwork and select **`Views > Lon Utilities Manager`**.

The **`Lon Utilities Manager`** appears.

Step 2 At the bottom, click the *lower-left* drop-down control (command menu) and select **`Identify`**.

Click the drop-down control to the *right* (command submenu) and select **Service Pin**.

Note: *For this command, selected LON device makes no difference ("LocalLonDevice", other).*

Step 3 Click the **Execute** button.

The view updates to show a timestamp with "Waiting on service pin" below.

Step 4 At any connected LON device, press its service pin.

In the utilities manager view, the device's domain table is shown, including the domain length (Domain Len) and ID for the device. If the device is configured on something other than the zero-length domain (Domain Len not "0"), you must configure the LonNetwork's working domain to match. See the next section, "To change the working domain".

To change the working domain

Note: *Niagara's working domain always uses domain index 0 (do not confuse with zero-length domain). If you change the default working domain, the zero-length domain is retained, but moved to domain index 1. If operating as network manager (typical), the network's Lon Netmgmt always needs access to the zero-length domain in order to receive service pin messages.*

To change the working domain for a station's LonNetwork, from the zero-length domain:

Step 1 Right-click the LonNetwork and select **Views > Property Sheet**. The property sheet appears.

Step 2 In the property sheet, click to expand **Lon Netmgmt** (see Figure 3-10 on page 9).

Step 3 In **Domain Id** property, click the **length** drop-down and select the matching domain-length used by other Lonworks devices (either 1, 3, or 6).

The adjacent **id** field changes from empty to showing bytes (for example, 00 or 00 00 00).

Step 4 In the **id** field, enter the domain id used by other devices. Domain IDs are hexidecimal bytes. For example, if domain length is 1, the ID range is from 00 to FF.

Step 5 Click the **Save** button.

The LocalLonDevice is now configured with a non-default working domain.

Note: *To place other (existing) Lon devices in this domain, you must recommission them.*

Discover and learn Lon devices

After adding the LonNetwork, you can use online discovery to find and create Lon devices under the LonNetwork. An available "**Quik Learn**" feature can greatly simplify this. You use the LonNetwork's default **Lon Device Manager** view.

This section provides quick start procedures for both tasks, as follows:

- Using online Discover to see Lon nodes
- Using Quik Learn for a new LonNetwork

Note: *For general information, see the "About the Device Manager" section in the* Drivers Guide*. For specific details, see "About the Lon Device Manager" on page 3-14.*

Using online Discover to see Lon nodes

If the JACE is connected to the network of Lon nodes, this is the quickest way to check if devices are ready to be added to the station. Use the following procedures:

- To discover Lon nodes
  and if necessary,
- To see a node's current domain table

To discover Lon nodes

Perform this task to discover Lon nodes.

To discover Lon nodes:

Step 1 In the Nav side bar, double click the **LonNetwork** to bring up the **Lon Device Manager**.

Step 2 Click the **Discover** button to automatically learn what devices are on the working domain (or, if different, the zero-length domain).

The manager view changes to "learn mode" (two panes), with a top "Discovery" pane.
A progress bar appears at the top of the view, and updates as the Lon Discover job occurs.

Step 3 When the discovery job completes, discovered Lon nodes are listed in the top table. The *bottom* pane, "Database," is a table of nodes that are currently mapped into the Niagara station—initially, this table will be empty.

For a new LonNetwork, instead of "Adding" devices, you typically use **Quik Learn** to populate your LonNetwork. See "Using Quik Learn for a new LonNetwork" on page 2-3.

Note: *This works a little differently than the device manager in most other NiagaraAX drivers. See "Lon Device Manager key points" on page 3-14 for more details.*

Note: *If you do not see a node, it is likely configured on a different domain than either the LonNetwork's working domain or the zero-length domain. To see such a node, press its Lonworks service pin. It should now appear listed in the Discovery pane.*

To see that node's current address configuration (including domain), see the next section "To see a node's current domain table".

To see a node's current domain table

You can do this for any Lon node, including any node where you have just pressed its service pin.

Step 1 Right-click the LonNetwork and select **Views > Lon Utilities Manager**.

The **Lon Utilities Manager** appears.

Step 2 Near the bottom, click the (device) drop-down and select the node just found from a service pin.

Step 3 Click the *lower-left* drop-down control (command menu) and select **Data Structs**.

To the *right* (command submenu), click the drop-down control and select **Domain Table**.

Step 4 Click the **Execute** button.

The view updates to show the device's domain table, including the domain length (Domain Len) and ID for the device. If the device is configured on something other than the zero-length domain (Domain Len not "0"), you must configure the LonNetwork's working domain to match. See "To change the working domain" on page 2-2.

Using Quik Learn for a new LonNetwork

Typically, you use the "**Quik Learn**" (spelling intentional) function in the Lon Device Manager to populate your new LonNetwork with the proper LonDevice components. This function is useful for both previously "managed" and "unmanaged" Lonworks networks, further defined as follows:

- A previously *managed* network means that all nodes were already installed and commissioned using a Lonworks network management tool. Each device has a unique subnet-node address, and usually coherent Lon bindings between other devices. You can preserve this configuration using **Quik Learn**. See "To populate a new network from previously managed network" on page 2-3.
- An *unmanaged* network means that nodes were installed using default subnet-node addresses, where there are often duplicate addresses or other address conflicts. See "To populate a new network from previously unmanaged network" on page 2-4.

To populate a new network from previously managed network

Perform this task to populate a new LonNetwork in your station database, where existing Lon network management is preserved.

To populate your new LonNetwork:

Step 1 Double-click the LonNetwork to bring up the **Lon Device Manager**.

Note: *If in split-pane (learn mode), click the Learn Mode tool to toggle out of learn mode.*

Step 2 Click the **Quik Learn** button.

A popup **Learn** dialog appears.

Step 3 In the Learn dialog, leave the *top option cleared*, and either check (or clear) the bottom options "`Learn Links: add links to the database`" and "`Upload Config: upload ncis and cps`", and click **OK**.

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

Step 4 When the learn job completes, learned Lon nodes are listed in the "Database" table. All devices should appear listed with unique subnet/node addresses.

- If you chose to learn links, you can see any learned bindings in other views of the LonNetwork, including the **Lon Link Manager** view or (graphically) in the network's wire sheet view.
- If you chose to upload config, you can see the values of any device's ncis from its **Nc Manager** view (right-click the device and select **Views > Nc Manager**).

Step 5 You can create Lon proxy points under any device. See "Create Lon proxy points" on page 2-7.

To populate a new network from previously *unmanaged* network

Perform this task to populate a new LonNetwork in your station database, where Niagara applies unique and contiguous subnet-node addressing to learned devices.

To populate your new LonNetwork:

Step 1 Double-click the LonNetwork to bring up the **Lon Device Manager**.

Note: *If in split-pane (learn mode), click the Learn Mode tool to toggle out of learn mode.*

Step 2 Click the **Quik Learn** button.

A popup **Learn** dialog appears.

Step 3 In the Learn dialog, check "`Unmanaged Network: ignore subnet/node conflicts`," and (typically) "`Upload Config: upload ncis and cps`", and click **OK**.

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

When the learn job completes, learned Lon devices are listed in the "Database" table.

Step 4 Click to select all devices, and click the **Commission** button.

A progress bar appears at the top of the view, and updates as the Lon Commission occurs. See "About Commission" on page 3-15 for more details.

Step 5 When the commission job completes, all devices should appear listed with a "`Config Online`" state, and with unique subnet/node addresses.

You can create Lon proxy points under any device. See "Create Lon proxy points" on page 2-7.

Create offline database and match

If you know the Lon device types you will have under your LonNetwork, you may wish to engineer the LonNetwork offline, or even manually add them when online. When adding a new device, you can select a device component from a specific `lon<Vendor>` palette.

Note: *Each* `lon<vendor>` *module contains a palette with a device for each Lon Xml (*`.lnml`*) file in that module. As needed, you can simply drag and drop devices from an open palette into the LonNetwork. See "To drag and drop devices from a lon<Vendor> palette".*

Alternatively, you can select a **DynamicDevice** from the lonworks palette and specify an lnml (Lon Xml) file that identifies the particular device type. A device is associated with a particular lnml file by its Lonworks program ID.

Later, when online with the Lon network, you can use the Discover and *Match* feature in the Device Manager to map each device object with a particular discovered Lon node. A match is possible only if the program IDs are the same. The match synchronizes the Lonworks neuron Id and applies appropriate subnet-node addressing.

For general information about matching, see the "Match (Device)" section in the *Drivers Guide*. Note that the Lonworks match offers an address source *toggle*, which you select depending on whether the network is already managed or unmanaged. See "Lon Device Manager key points" on page 3-14.

The following procedures explain how:

- To drag and drop devices from a lon<Vendor> palette
- To add a DynamicDevice
- To import Lon Xml into DynamicDevice
- To match a Lon device to a previously managed device
- To match a Lon device to a discovered unmanaged device

To drag and drop devices from a lon<*Vendor*> palette

As noted above, you can simply drag and drop devices from the opened palette of any `lon<Vendor>` module, as described below.

Step 1 In Workbench, open the **Palette** side bar, if not already open.

Step 2 Open any of the `lon<Vendor>` palettes with devices of interest, for example, `lonHoneywell`.

Available devices are listed by default names matching the source `.lnml` file name.

Step 3 Open the target view of the target **LonNetwork**, or of the target **Lon Device Folder** (under the LonNetwork). This may be the device manager view or the wire sheet view, for example.

Step 4 Drag a device from the palette into the target view (or if desired, from the palette onto the LonNetwork node in the Nav tree).

A popup **Name** dialog appears, in which you can enter a meaningful name.

Step 5 Click **OK**.

The device is added to the station database.

Note: *At the time of this document, you should also perform additional steps below if programming a station in a remote JACE, and it does not have the same* `lon<Vendor>` *module already installed (as the local module palette that you are copying from). At some future time, these additional steps may not be needed.*

Step 6 Right-click the newly-copied device, and on its popup menu select the ImportXml command.

The **ImportXml** dialog appears, showing "`module://lon<Vendor>/<device>.lnml`", corresponding with the device you just dragged from the palette.

Step 7 Click **OK**.

Data is read from the local `.lnml` file into the device component in the remote station. The device is now ready for offline programming, or if online with the JACE (and LON network), ready for matching with an actual device.

See "To match a Lon device to a previously managed device" on page 2-6 or "To match a Lon device to a discovered unmanaged device" on page 2-6.

To add a DynamicDevice

You can add a DynamicDevice using the **New** button in the Lon Device Manager, as described here.

Note: *You can also simply drag and drop a DynamicDevice from the* **lonworks** *palette, naming it as you wish. In either case, after adding it, an extra task is required to make it useful. See "To import Lon Xml into DynamicDevice" on page 2-5.*

To add a DynamicDevice using the New device wizard:

Step 1 In the Lon Device Manager, click the **New** button.

This brings up the New device wizard dialog.

- In "Type to Add," select **Dynamic Device**.
- Enter the number of devices to add (default value is `1`).
- Click **OK**.

Step 2 When the next **New** dialog appears, edit the name of the component as you wish to see it in the Niagara station.

Step 3 For purposes here, leave all values at defaults and click **OK**.

The component is added to the station, listed in the Database pane as type "`DynamicDevice`."

Note: *In the New device wizard, alternate "Types to add" may appear beside "Dynamic Device," for example, "Q7300," "Xl10 Chc1" and "Xl10 Hyd2." These are special subclasses of a LonDevice. If you select one of these, its Lon Xml (lnml) file association is already made.*

To import Lon Xml into DynamicDevice

Step 1 Right-click the **DynamicDevice**, and on the popup menu select the ImportXml command.

The **ImportXml** dialog appears, showing "`null`" for the Xml File (if not previously specified).

(Alternatively, you can also click the folder icon beside the **Lon Xml File** entry in any device's **Add** or **Edit** dialog, for the File Chooser, and continue with step 3 below.)

Step 2 Click the folder icon for the **File Chooser** dialog. The File Chooser opens.

Step 3 In the **File Chooser**, click to navigate to the location of the lnml file needed, for example a "jar'ed" lnml file in a lon<*vendor*> module jar: (example: `My Modules`, `lonSiebe`). Or, another lnml file previously made using the Lon Xml Tool.

Step 4 If selecting a "jar'ed" lnml in the **File Chooser**, double-click that lon<*vendor*> jar.

Its available lnml files are listed by name, for example: `Mnlrh2.lnml`, `Mnlrh3.lnml`, and so on. (Each lnml file name corresponds to the known `.xif` file name for any device.)

Step 5 Click an lnml file, then click **Open**.

The **File Chooser** closes, and the Import Xml dialog shows the selected lnml file.

Step 6 Click **OK** to import the lnml file into the DynamicDevice component.

Now, the component lists showing a type other than DynamicDevice, for example, "`Mnlrh3`."

After you do this, you can use the **Lon Point Manager** under that device to add (offline) proxy points, because data items are now known. See "Create Lon proxy points" on page 2-7.

To match a Lon device to a previously managed device

If online with the Lon network, you can *match* a manually added Lon device to a discovered node using the Device Manager's **Match** button. This copies address data (learned in the discovered device) into the selected Lon device component. See "Lon Device Manager key points" on page 3-14.

To match an existing DynamicDevice to a discovered (previously managed) node:

Step 1 If the Discovered node table is empty, perform a Discover. See "To discover Lon nodes" on page 2-2.

Step 2 Click to select (highlight):

- *One* device in the *Discovered* table.
- *One* device in the *Database* table.

Step 3 Click the drop-down control on the **Match** button to select the following:

- Use Net Subnet/Node — This specifies to upload address data from the node.

Step 4 Click **Match**.

This brings up the **Match** dialog. As needed, edit any property required. For specific details, see "About Lon devices" on page 3-29.

Step 5 Click **OK** to match the discovered device to the DynamicDevice component.

The device is removed from the Discovered table, and remains selected in the Database table.

Step 6 Right-click the device, and select **Actions > Upload**.

In the popup **Upload** dialog, check all (recursive, transient, persistent).This retrieves current configuration data from that device.

Step 7 Repeat this process (steps 2 through 6) for each previously-managed device.

Step 8 When all devices are matched, click the Learn Mode tool to toggle out of learn mode.

Step 9 Click to select (highlight) all the devices just matched.

Step 10 With the matched devices selected, click the **Quik Learn** button.

A popup **Learn** (links) dialog appears. Click **OK**.

A learn links job is launched, where Lon bindings are learned.

Step 11 When the learn links job completes, all devices should appear listed with a "`Config Online`" state, and with unique subnet/node addresses.

To match a Lon device to a discovered unmanaged device

If online with the Lon network, you can *match* a manually added Lon device to a discovered, unmanaged node using the Device Manager's **Match** button. This copies address data assigned by Workbench into the selected Lon device component. See "Lon Device Manager key points" on page 3-14.

To match an existing DynamicDevice to a discovered (previously unmanaged) node:

Step 1 If the Discovered node table is empty, perform a Discover. See "To discover Lon nodes" on page 2-2.

Step 2 Click to select (highlight):

- *One* device in the *Discovered* table.
- *One* device in the *Database* table.

Step 3 Click the drop-down control on the **Match** button to select the following:

- Use Db Subnet/Node — This specifies automatic assignment of subnet/node address.

Step 4 Click **Match**.

This brings up the **Match** dialog. As needed, edit any property required. For specific details, see "DynamicDevice properties."

Step 5 Click **OK** to match the discovered device to the DynamicDevice component.

The device is removed from the Discovered table, and remains selected in the Database table.

Step 6 Right-click the device, and select **Actions > Upload**.

In the popup **Upload** dialog, check all (recursive, transient, persistent). This retrieves current configuration data from that device.

Step 7 Repeat this process (steps 2 through 6) for each previously-unmanaged device.

Step 8 Click to select all `Unconfigured` devices, and click the **Commission** button.

A progress bar appears at the top of the view, and updates as the Lon Commission occurs. See "About Commission" on page 3-15 for more details.

Step 9 When the commission job completes, all devices should appear listed with a "`Config Online`" state, and with unique subnet/node addresses.

Create Lon proxy points

As with device objects in other drivers, each Lon device 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 **`Lon Point Manager`**). You use it to add Lon proxy points under any LonDevice.

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

Note: *The* **`Lon Point Manager`** *works differently than in other driver's Point Manager views. For example, when toggled out of "learn mode" (Database pane only), the* **`Add`** *button is unavailable for "manually" adding proxy points. Candidates for Lon proxy points are always determined by the set of network variables (nvs, ncis) associated with that device type. Once you learn the device, this information is then known to Niagara, and available in learn mode. If programming offline, this information is also known once you specify a Lon Xml File (.lnml) in the parent LonDevice. In either case, there is no "point discovery job" for specific data items in any Lon device.*

To create Lon proxy points

Once you learn a LonDevice, or create it offline and assign it a specific Lon Xml File, you can add Lon proxy points under its Points extension. Use the following procedure:

To create Lon proxy points in a device:

Step 1 In the **`Lon Device Manager`**, in the **`Exts`** column, double-click the `Points` icon in the row representing the device you wish to explore.

This brings up the **`Lon Point Manager`**.

Step 2 Click **`Discover`** to make sure the view is in split-pane (learn mode).

The device's network variables (nvis, nvos, and ncis) are listed in the *top pane* of the view, in the "Discovered" table. Initially, each network variable (nv) occupies one row. Often, an nv is implemented with a structured SNVT (multiple data fields), and so is listed with a "+" you can expand to see other data fields. (By default, the "first" data field in the SNVT structure appears on top.) Each row in the Discovered table represents one Lon proxy point candidate.

See "Lon Point Manager tips" on page 3-38 and "Lon proxy point type selection" on page 3-39 for more details.

Step 3 Click to select the data items you wish to proxy. Often, for an nv implemented as a structure, you may need one or more secondary data fields instead of (or in addition to) the first "top" data field.

Step 4 You can map selected items in the station in a number of 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).
- Click to select in Discovered, then press "a". ("Quick Add", meaning *no* **`Add`** dialog).

This works the same as in other driver's Point Manager views.

Step 5 When the **`Add`** dialog appears, you can edit the configuration of each proxy point's LonProxyExt before it is added in the Niagara station. Initial property values are determined by Niagara, based upon how the nv or nci was implemented in the device.

Note the following about entries in the Add dialog:

- **`Name`** is the "nvi, nvo, or nci name", and if part of a data structure, a trailing "_elementName" to ensure a unique point name. For example: "`nciTempSetPts_occupiedCool`"
  This is the Niagara point name only—change if needed (does *not* affect Lonworks node).
- **`Type`** is the Niagara control point type to use for the proxy point. For nvis and ncis, the default selection is a "writable" control point; whereas nvos can be "read-only" points only.
  Note: *Unlike other editable entries in the Add dialog, you cannot edit Type later.*
- **`Target`** is the LonComponent represented by the proxy point, for example "nvoStatus."
- **`Element`** is the data element if a structure, or the SNVT type if an unstructured item
- **`Facets`** represent the parent Niagara proxy point's facets, for how the value should be displayed in Niagara.
- **`Conversion`** specifies the conversion to use between the "read value" (in Device Facets) and the

parent point's facets, where "Default" is typically used.

- **Link Type** specifies the Lon service type to use when binding to this item.

Step 6 When you have Lon proxy point(s) configured properly for your usage, click **OK**.

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

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

Note: *In some cases, you may need to edit a point's facets or even conversion type. For related details, see the following topics:*

- "Facet conversion in Lon proxy points" on page 3-4
- "Differential temperature notes" on page 3-4
- "Facet support for "null" output" on page 3-5

Bind Lon proxy points

By default, when you add Lon proxy points, the source LonComponent *nvs* are all *polled* for values. For many proxy points, particularly nvos, this is less efficient than binding (from the source device) to the target LocalLonDevice, such that Lonworks nv updates are received.

Once you have added Lon proxy points under one or more devices, you typically bind to receive nv updates. The following procedures explain how:

- To bind all Lon proxy points
- To bind some Lon proxy points

To bind all Lon proxy points

Typically, you bind to *all nvs* used for Lon proxy points, unless there is a specific reason not to do this. One possible scenario may include devices that have nvos that send updates too frequently, and there is no way (in such devices) to adjust the "minimum write time" upwards.

Note: *In that example scenario, even after a global* **Bind** *command in the* **Lon Link Manager**, *you can specifically pick nvs (in srcDevices) to change to "Poll Only."*

To bind Lon all proxy points, use the following procedure:

Step 1 In the Nav side bar, right-click the LonNetwork and select **Views > Lon Link Manager**.

This brings up the Lon Link Manager.

Each newly-added proxy point appears listed as a row in the NetworkVariableLinks tab, with a linkStatus of "NewLink." By default, each point uses a linkType of "Standard."

Step 2 Click the Bind button to invoke a **Lon Bind** job.

A progress bar displays at the top of the manager, and the bind job finishes. Now, the linkStatus of entries for proxy points should show as "Bound."

See "About the Lon Link Manager" on page 3-20 for more details.

To bind some Lon proxy points

You can selectively bind to only some nvs used for proxy points, and leave others at the current link status (including "NewLink").

To bind only some Lon proxy points, use the following procedure:

Step 1 In the Nav side bar, right-click the LonNetwork and select **Views > Lon Link Manager**.

This brings up the Lon Link Manager.

Each newly-added proxy point appears listed as a row in the NetworkVariableLinks tab, with a linkStatus of "NewLink." By default, each point uses a linkType of "Standard."

Step 2 If needed, click on the **srcDevice** column header to resort the NetworkVariableLinks table by devices (this groups nvo sources together by Lon device name).

Step 3 In the NetworkVariableLinks table, click to select the nvs for proxy points you wish to bind.

Step 4 Click the Selective Bind button to invoke a **Lon Bind** job.

A progress bar displays at the top of the manager, and the bind job finishes. Now, the linkStatus of entries for the selected items should show as "Bound."

See "About the Lon Link Manager" on page 3-20 for more details.

Chapter 3

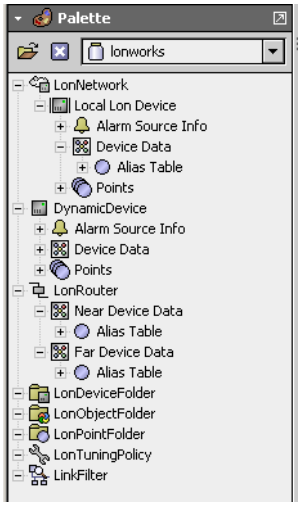
Niagara Lonworks Concepts

This section describes the NiagaraAX Lonworks implementation. These are the main subsections:

About lonworks palette components

In Workbench, open the `lonworks` palette in the Palette side bar to examine various lonworks (Lon) components, as shown in Figure 3-1.

Figure 3-1 *Expanded lonworks palette*



As with most other NiagaraAX drivers, you *rarely* need to work from the palette. Instead, the various Lon manager views simplify component creation, enforcing proper component hierarchy.

One exception is the LinkFilter component, a *debug-type* object that provides "low-level messaging" related to Lon links via its table-based (default) **`Link Filter View`**. Application is intended for troubleshooting only. To use it, you copy the LinkFilter from the lonworks palette into the LonNetwork container (component at device level). The LinkFilter component also has "tuning" properties that affect what displays in the Link Filter View. See "lonworks-LinkFilterView" on page 4-1 and "lonworks-LinkFilter" on page 5-5 for related details.

Note: *An additional* **`kitLon`** *palette is available, to supplement the lonworks palette. At the time of this document update, kitLon contains five components:*

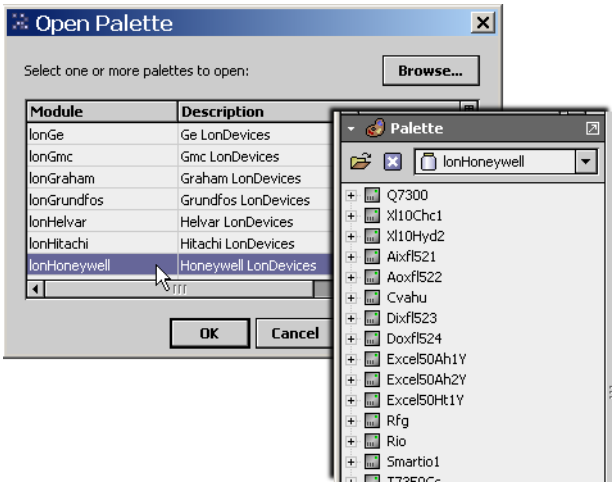
- **BufferParams**
  (AX-3.5 or later) A component you can copy into your **`LocalLonDevice`** for configuring Neuron chip message buffers on the JACE's Lonworks interface. For highly knowledgeable users only!

- **LonTime**
  Uses "system time", and can be linked to a Lon device's nvi that uses `SnvtTimeStamp`.
- **LonTodEvent**
  Can take inputs from BooleanSchedule outputs (Out, Next Value), and be linked to a Lon device's nvi that uses `SnvtTodEvent`.
- **LonPoint**
  (AX-3.3 or later) Allows you to create a control object that has an nvo with a selectable SNVT type, which can be used to link to an nvi on one or more Lon devices.
- **LonReplace**
  (AX-3.4 or later) Allows you to issue a Replace command (action) on a Lon device from a Px page binding to this component placed under the parent Lon device, rather than requiring Workbench and Lon Network Manager view to perform a Replace on a device.

As needed, copy objects from the kitLon palette into the LonNetwork, for use in special applications.

Note: *Also, each* `lon<Vendor>` *module includes its own palette, which you can open in Workbench's palette side bar, and manually copy devices into a LonNetwork.*

Figure 3-2 *Opening specific lonVendor palette*



See "To drag and drop devices from a lon<Vendor> palette" on page 2-4 for a related procedure.

Understanding network management scenarios

Lonworks network management requires a "single owner" to perform and maintain an accurate database of things such as node address assignments, nv bindings, message services used, and a raft of other definitions. The Niagara lonworks driver *supplies* this network management capability, by default. Typically, you engineer a LonNetwork with the station acting as the "network manager" for all Lonworks devices on that network. See "Station as network manager" on page 3-2.

However, in some scenarios you may wish to install a JACE as simply another "Lonworks node," where another (*external*) Lonworks network management tool is used (for example, LNS). In this case, you configure the LocalLonDevice to *not* perform network management. Instead, you expose other station data as network variables (nvs) and ncis under the LocalLonDevice.

In this application, you *do not* use the various views of the LonNetwork (Lon Device Manager, Lon Link Manager, and so on) as described in much of this document. For more details, see "Notes when configuring as Lon node" on page 3-40.

Station as network manager

In this typical application of the lonworks driver, you use Workbench to "model" all other Lon devices in the station's database (under the LonNetwork). The lonworks driver provides online "learning" capabilities to simplify this. Then (as needed), you also model data in each device using Lon proxy points.

The various manager views on the LonNetwork provide your interface to all network management operations performed by the station. This includes all engineering, maintenance, and troubleshooting of the Lonworks network (including all bind operations). In addition, you can establish links directly between Lon nodes, typically by using the wire sheet view of the LonNetwork.

Depending on the installation scenario, you may need to perform a learn of a network of Lon devices that are already configured (previously managed), or, on a previously-unmanaged (new) network. Both learn types are supported, however, engineering considerations apply.

See the following sections:

- About an unmanaged network
- About a previously managed network

About an unmanaged network

On an unmanaged network, Lonworks devices are often unconfigured (not addressed, or operating with identical addresses), and Lon bindings between devices do not exist. There is no existing Lon network management to learn (only to establish for the first time).

About a previously managed network

On a previously managed network, Lonworks devices are uniquely addressed and configured, often with existing Lon bindings in use between devices. In this case, you usually want to learn all existing network management.

Note: *At this point forward, the station (Niagara) must become the only Lonworks network manager. Otherwise, proper network management cannot be successfully maintained.*

Notes on unit conversion

Note: *This section applies to U.S. installations only.*

Lonworks devices use SNVTs and SCPTs to exchange data, which typically use International System of Units (SI), such as degree Celsius °C for temperature, l/s (liters per second) for volumetric flow, and so on. Consider these the "native" units, in which numeric values are exchanged between devices.

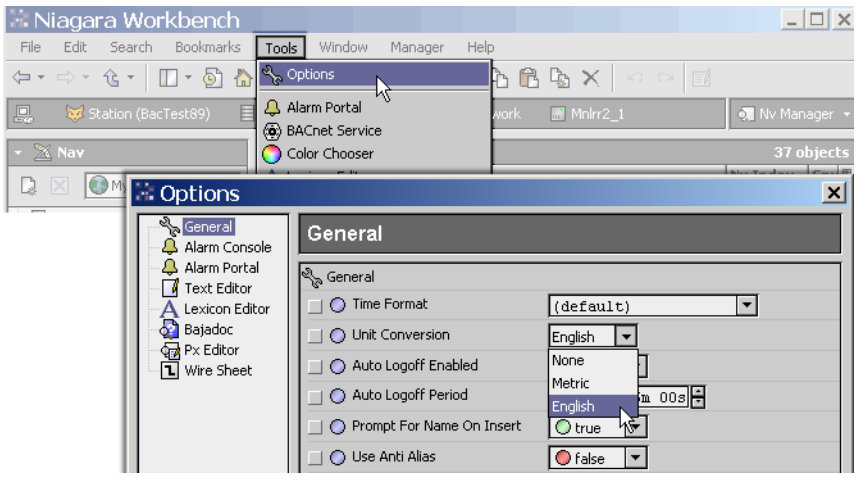
In U.S. installations (only), you typically want to convert "metric" values to "English" values for the user interface, such as degree Fahrenheit °F for temperature, cfm (cubic feet per minute) for volumetric flow, and so on. If engineering a U.S. installation of a LonNetwork, it is important to understand the available unit conversion options, described in the following sections:

- Workbench (display) unit conversions
- Facet conversion in Lon proxy points

Workbench (display) unit conversions

In the Workbench program running on your PC, you may globally set unit conversion to automatically *display* "Out" values in English units (where the default is *no conversion* for display). Do this from the menu selection **`Tools > Options > General`**, as shown in Figure 3-3.

Figure 3-3 *Unit conversion (display) in Workbench*



With Workbench unit conversion set to English, out values (for example, visible in a Lon device's **`Nv Manager`** view, or within LonData in a device's LonComponents) automatically display with the expected English units and corresponding numeric values. For related topics, see "About the Nv Manager" on page 3-31 and "Using LonData directly" on page 3-37.

Note: *This setting is local to your Workbench display usage only! If providing other users browser access to LonData (or any data using SI units), and need the same automatic display conversion, you must specify English unit conversion in each of those station* **`User`** *accounts (Facets, Unit Conversion). For more details, see the "User Manager" section in the* User Guide*.*

In addition, this does not take the place of adjusting Facets in Lon proxy points. See the next section "Facet conversion in Lon proxy points" on page 3-4 for more details.

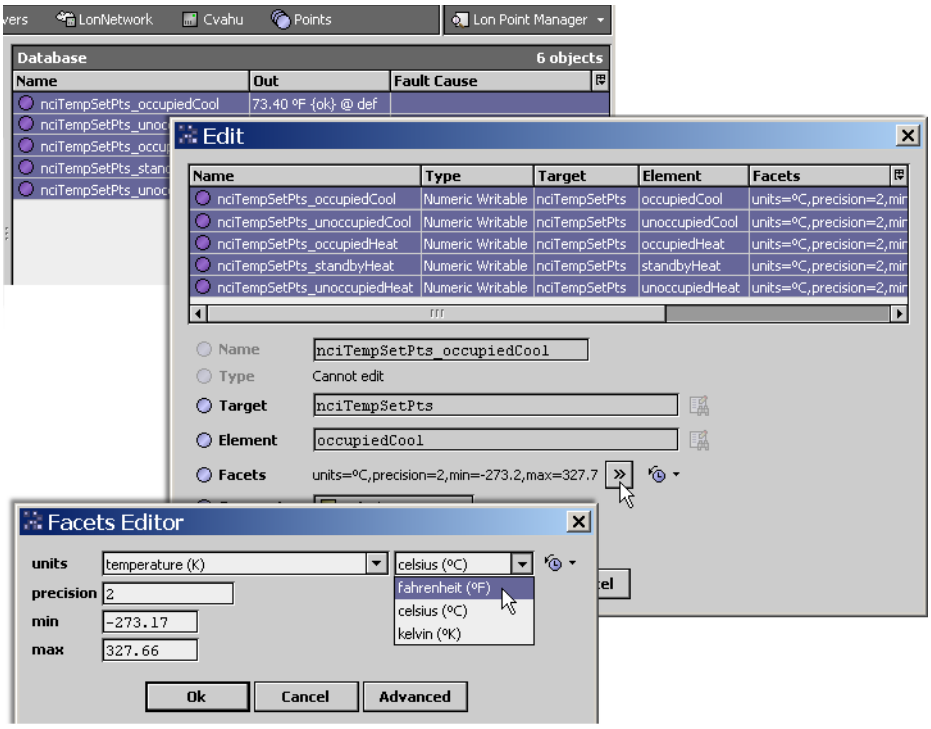
Facet conversion in Lon proxy points

When you create Lon proxy points under a Lon device, by default, each parent point's facets are the same as the "device facets" in its Lon ProxyExt (SI units).

Note: *This occurs regardless of your Workbench "option" settings for unit conversion display. See "Workbench (display) unit conversions" on page 3-3 for more details.*

For example, if you add Lon proxy points for a device's nci "nciTempSetPts", for each data element (using SNVT_temp_setpt), each of the corresponding Lon proxy points default to facets of °C.

Figure 3-4 *Default point facets reflect device facets*



In U.S. installations, you typically edit the points' facets to English equivalents. Otherwise, without any further change, as shown in Figure 3-4 top, (with your Workbench option set to English unit conversion), point "Out" values *display* in English units (such °F) in the Lon Point Manager. However, the *actual* (numeric) Out *values* remain in SI units (such as °C). If you link point outputs to other components, and/or add history extensions to these proxy points, the SI unit values are used.

Therefore, in a U.S. installation you should set the facets of Lon proxy points to the needed English equivalents, to avoid any control issues or other confusion. As shown in Figure 3-4 middle and bottom, you can do this with a "gang edit" of multiple (related) proxy points from the Lon Point Manager, or even in the initial **`Add`** dialog when you first create the proxy points.

Differential temperature notes

In cases where proxy points are added for nvs or ncis known to have a "temperature differential" (delta) application, yet the point is using "absolute temperature" facets/units (e.g "fahrenheit"), you should change the proxy point's facets/units to differential type, i.e. "degrees fahrenheit." This prevents issues with unit conversions performed by Niagara. This also modifies the proxy units, device units, and the element qualifiers in the LonPrimitives in the associated Lon Component.

Note that you can also edit the Lon Xml (lnml) file for a device, if needed, to accomplish the same thing. See "Differential temperatures and lnml file edits" on page A-4.

Facet support for "null" output

Starting in AX-3.5 (also builds 3.3.32 and later, and 3.4.54 and later), Lon proxy point support was added for values received from numeric and enumerated LonData elements with "invalid" or other specified values to be represented as "`null`" values, versus the "`nan`" (not a number) representation for invalid.

This can be useful if the point's out slot is linked to the input of a math component (an Average object, for example), or to the priority array input of a writable control point. The null value causes the value to be ignored, instead of evaluated as invalid. In the linked Average example, its output reflects the average of the other "non-null" input values (instead of displaying "nan"). The linked writable point will drop down to the next-highest priority arrayed input (or fallback) for control, instead of also displaying "nan".

To preserve backward compatibility, support requires that you edit the proxy point's facets and *add* an additional entry, specifying a Key, Type, and Value. Depending if Numeric or Enum, the following facets are supported:

- **Numeric**
  - Key: `nanIsNull` - Type: `Boolean` - if Value = `true`, then invalid values will give null value
  - Key: `isNull` - Type: `Double` - if specified Value is present, then will give null value
- **Enum**
  - Key: `isNull` - Type: `Integer` - if specified Value is present, then will give null value

In the **`Config Facets`** dialog when editing a point's facets to add an entry, you must directly type in the needed Key value (not available in the Key drop-down list). See Figure 3-5 and Figure 3-6 for an example of adding the "`nanIsNull`" key for a numeric.

Figure 3-5 *Editing facets in Config Facets dialog (for proxy point to support null data)*

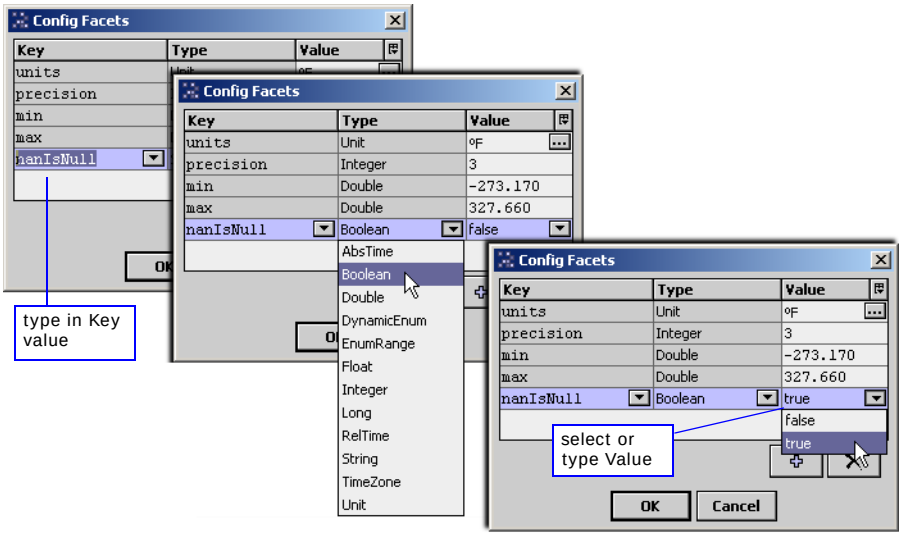
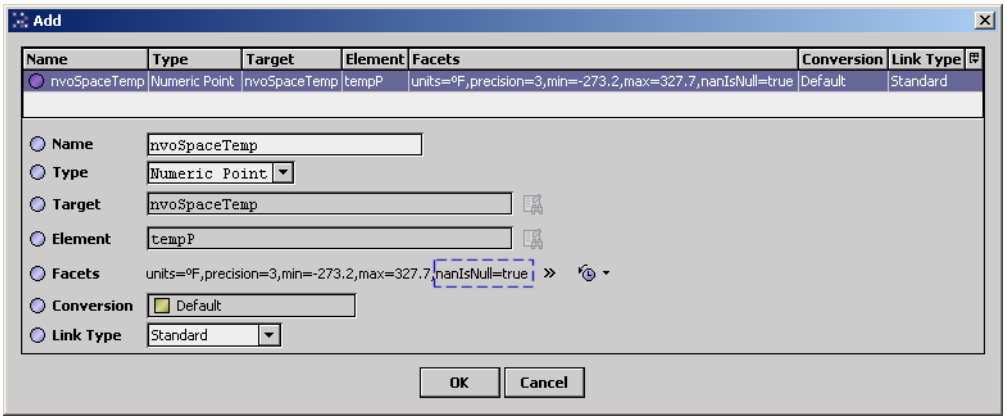


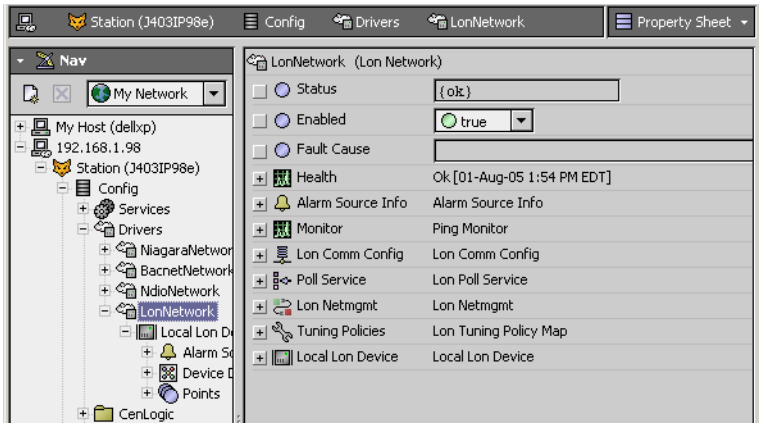
Figure 3-6 *Edit dialog for Lon proxy point where the facet above (nanIsNull) was added*



About Lon Network Architecture

Lonworks uses the standard NiagaraAX network architecture. See "About Network architecture" in the *Drivers Guide* for general information. From the *property sheet* of the **LonNetwork**, you have access to all the major network-level container slots, as shown in Figure 3-7.

Figure 3-7 *LonNetwork property sheet*



Related topics include:

- LonNetwork status and monitor notes
- Lon Comm Config notes
- Lon Poll Service notes
- About Lon Netmgmt
- About the Local Lon Device

LonNetwork status and monitor notes

Status of a LonNetwork is typically "ok" or perhaps "fault" (fault results if lonworks feature is not licensed). The "Fault Cause" property further explains any fault status. A down status occurs if you manually set the network's Enabled property to `false` (from the LonNetwork's property sheet). The **Health** slot contains historical timestamp properties that record the last network status transitions from ok to any other status.

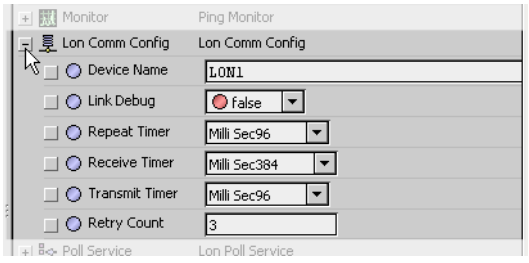
Note: *As in other driver networks, the LonNetwork has an available "Alarm Source Info" container slot that you can use to differentiate a LonNetwork alarm from other component alarms in the station. See "About network Alarm Source Info" in the* Drivers Guide *for details.*

The network's **Monitor** component verifies the presence of networked devices. It sequentially pings all devices that are mapped in the station. For more details, see "About Monitor" in the *Drivers Guide*.

Lon Comm Config notes

In the property sheet of a LonNetwork, the Lon Comm Config container holds properties needed to configure the communication stack for a single Lonworks connection.

Figure 3-8 *Lon Comm Config properties of LonNetwork (default values)*



Typically, you leave all Lon Comm Config properties at default values, as shown in Figure 3-8. The exception is when you have multiple LonNetworks (physical Lon ports), where each LonNetwork must have a unique Device Name (LON1, LON2, etc.).

Note: *Enabling* **Link Debug** *will show the raw byte data for messages from/to the Neuron. Note that you can also enable link debug from the "Spy" page access of the station (e.g. right-click station:* **Spy > logSetup**, *enable* `Trace` *on* `lon1`*).*

Refer to the LonWorks Host Application Programmer's Guide *(Appendix C), for message header meaning (first 16 bytes). Refer also to the* Neuron C Data Book *(Appendixes A and B), for definition of LonTalk messages.*

Lon Comm Config timers and retry count are used on outgoing messages for all network management messages, all poll messages, and unbound NV updates messages from the station.

Note: *Sometimes, certain Lon devices may process Lon messages "slower than normal," resulting in errors when you do commissioning or binding operations--where the commission or bind reports as failed. You can confirm such problems by using the Lon Utilities Manager (afterwards) and running a* **`verify`** *report. The verify report will list the inconsistencies between the bindings on the devices, and the list of bindings that Niagara's LON network management determines that they should have.*

In an example with a Lon network containing such "slow responding" devices, operation may improve with Lon Comm parameters adjusted upwards to:

- Repeat Timer: mSec128
- Receive Timer: mSec1024
- Transmit Timer: mSec128
- Retries: 4

In this case, it may also be necessary to adjust "link descriptors" under the Lon Netmgmt container from default values. See "Link descriptor notes" on page 3-10.

Lon Poll Service notes

As a container slot under the LonNetwork, the LonPollService operates as in most other drivers. See "About poll components" in the *Drivers Guide* for general information. In a LonNetwork, polling provides one means to update station database values from transient nv data in networked Lon devices.

Note: *Polling is used when proxy points are first created, before you bind nvs to the station (LocalDev) using the Lon Link Manager. Typically, you bind nvs so that most updates occur using nv updates, vs. polling requests. As necessary, polling also occurs to update nv values of LonComponents (visible as nvis and nvos in the property sheet of a LonDevice). These exist even if proxy points are not created (see "About LonComponents" on page 3-35).*

The LonPollService sequentially poles all the devices on the network, and reads nv data per the nv update rules. See the next section, "Network variable (nv) poll/update rules"

Network variable (nv) poll/update rules

General rules for polling nv (nvi and nvo) values are as follows:

1. An nv is subscribed if it is being viewed or has a proxy point which is subscribed. A proxy point is subscribed if it is viewed, or if it is linked to control logic (or has an alarm or history extension).
2. When an nv is first subscribed, its value is read (if device is in a valid state). Further reads are handled by the poll thread. (The poll service must be enabled for polling to function normally.)
3. Adding a proxy point creates a local connection to the associated nv. If a proxy point for an nvo, or a writable proxy point for an nvi, each appears as one entry (row) in the Lon Link Manager.
4. There is a "Bound To Local" property on nvProps which is set when an output nv with a proxy is bound. An output nv is not polled when it is bound locally. Instead nv updates will be received from the device.
5. The "Poll Enable" flag is set if there are unbound links and the nv is not bound locally. To prevent polling on an nv, the user may set its Poll Enable flag to `false`. The initial read is still performed when the nv is first subscribed, but subsequent polling by the poll service does not occur.

Polling rules for nvs with specific connection scenarios are provided in Table 3-1.

Table 3-1 *Polling rules*

Type of nv, connection	Update rules
nvi with writable proxy	Poll when nv or proxy point subscribes. If proxy value changes update nv.
nvi with read-only proxy	Poll when nv or proxy point subscribes.
nvo with writable proxy	<Not allowed>
nvo with read-only proxy (unbound)	Poll when nv or proxy point subscribes.
nvo with read-only proxy (bound)	Receive nv updates (*No polling*).
polled nvo* with read-only proxy bound (*see "Polled network variables")	Poll when nv or proxy subscribed.
nvi linked	Poll when subscribed.
nvo linked unbound	Poll when subscribed. If nvo changes push value to linked nvi. Write nvi value to device.
nvo linked bound	Poll when subscribed.

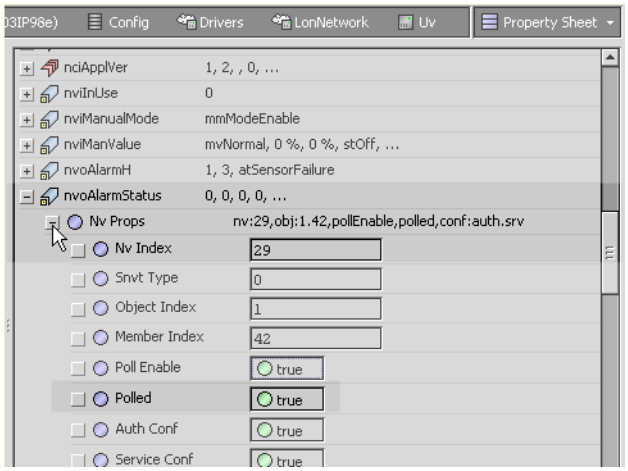
Note: *Input nvs (nvi) with read-only proxy points are never bound, and do not show up in the Lon Link Manager.*

Polled network variables

Some Lonworks devices include one or more nvos that are "polled" types, meaning that the nvo does not send nv updates upon a value change. Such an nvo must be polled. Typically, any such nv represents a data item that does not frequently change.

In the Lon device object, any such nvo LonComponent appears with its read-only "Polled" property set to "True" (as found under its "Nv Props" container). See Figure 3-9 for an example.

Figure 3-9 *LonComponent example of "polled only" nvo*



If you create a proxy point for such an nvo and bind to it in the Lon Link Manager, it still polls to receive value updates. Also, when creating a link (bind) between two Lon devices, the Link Editor only allows "polled" outputs to be linked to "polled" inputs.

Lon Tuning Policies

As a child of the LonNetwork, the LonTuningPolicyMap (Tuning Policies) operates as in most other drivers. See "About Tuning Policies" in the *Drivers Guide* for more details. By default, only a single LonTuningPolicy exists, however, you can add new tuning policies (duplicate and modify) as needed.

You assign a tuning policy to LonComponents (nvs) under a Lon device, *not* on specific proxy points. See "About LonComponents" on page 3-35. One way to do this is using the **`Nv Manager`** (default) view of a Lon device. See "About the Nv Manager" on page 3-31. Also see the next section, "Affects of tuning policy, by LonComponents".

Affects of tuning policy, by LonComponents

By LonComponent, the assigned tuning policy affects behavior as shown in Table 3-2.

Table 3-2 *Affects of tuning policy on LonComponent behavior*

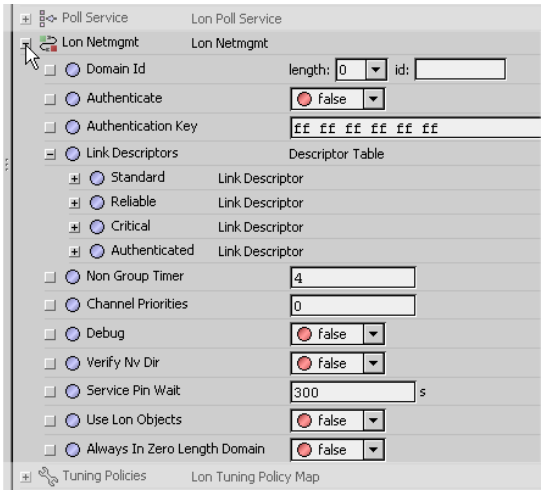
LonComponent	Affect
nci or cp	No affect
nvo	No affect
nvi with proxy (unbound)	Min/max times and start/on/enable flags apply
nvi with proxy (bound)	Min/max times and start/on/enable flags apply
nvi linked to nvo (unbound)	Min/max times apply to update
nvi linked to nvo (bound)	No affect

Note: *A writable proxy point only updates its target LonComponent when the control point is commanded to a new value, or when (in the case of an nvi) a write initiates because the maxSendTime was exceeded. Note it will not reassert (update) its "writeValue" if the device's value is changed by some other means, even if this is detected by a poll.*

About Lon Netmgmt

Lon Netmgmt is the base component for all Lonworks network management functions provided by the station. You perform most network management functions by using LonNetwork views. However, the LonNetmgmt component provides access to network management parameters, some of which you may need to modify in its property sheet. Figure 3-10 shows default Lon Netmgmt values.

Figure 3-10 *Lon Netmgmt container with default values*



Typically, you leave Lon Netmgmt properties at defaults, except for perhaps "Use Lon Objects." The following sections provide more details:

- Domain Id notes
- Service pin notes
- Use Lon Objects notes
- Link descriptor notes
- Always In Zero Length Domain notes

Domain Id notes

The default *working domain* for a Niagara LonNetwork (Domain Id in Lon Netmgmt) is the "zero-length" domain. Without further Niagara configuration, you can discover all Lon nodes that also belong to the zero-length domain. However, if you cannot discover nodes, they are likely configured on a different domain.

In this case, especially for a previously managed network where you wish to maintain current network management configuration (addresses, bindings) using **`Quik Learn`**, you must change the working domain of the Niagara LonNetwork to match. See the quick start procedures, "To identify a device using its service pin message" and "To change the working domain".

Service pin notes

In NiagaraAX, you do not have to expressly listen for a service pin message. In the **`Lon Device Manager`**, any unsolicited service pin message is handled as follows:

- If the node is *not* represented *in the database*, it is added to the Discovery table (learn mode).
- If the node *is* already *in the database*, it becomes *selected* (highlighted) in the Database table.

By default, when expressly listening for a service pin message from the Lon Utilities Manager view, you have 300 seconds of listen time for a service pin before the utilities manager cancels the wait period. This is configurable under the LonNetwork's Lon Netmgmt container.

Use Lon Objects notes

Starting in AX-3.3, the LonNetwork's Lon Netmgmt container includes a "Use Lon Objects" property, which by default is "false." See "LonObjects" on page 3-36 for more details.

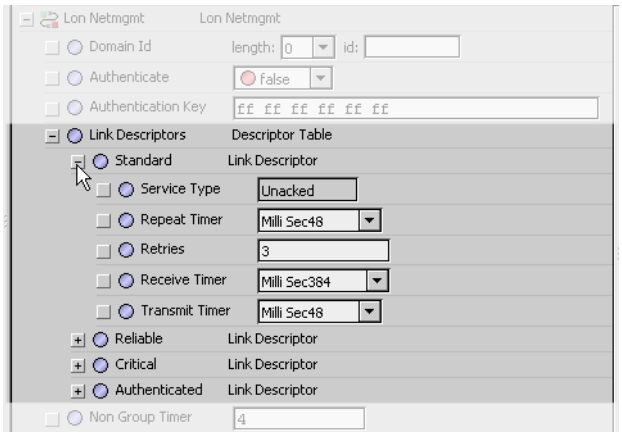
- While "false," any "Quik Learn" or "Add" device operates as in pre-AX-3.3 releases—where all of the learned LonComponents in the Lon device are placed directly under its device object.
- If set to "true," a Quik Learn or Add device results in an extra "LonObject" container grouping of the device's LonComponents. Each LonMark container represents a LonMark object.

Note that adding Lon devices with the LonObjects option affects the default **`Nv Manager`** view on a Lon device. For more details, see "About the Nv Manager" on page 3-31.

Link descriptor notes

Under Lon Netmgmt is a "Link Descriptors" container with four sub-containers, each associated with a bind service type (Standard, Reliable, Critical, Authenticate). Figure 3-10 shows the most commonly used bind service type (Standard), expanded with default link descriptor values.

Figure 3-11 *Standard service Link Descriptor properties (default values)*



Descriptors specify timers and a retry count applied to the address table of nodes during the bind process, and are used for all implicit addressing. Usually, you do not need to adjust link descriptor parameters from defaults.

Note that under each of the four types, the *default values* are the same:

- Repeat Timer: mSec48
- Retries: 3
- Receive Timer: mSec384
- Transmit Timer: mSec48

In cases of a Lon network that has "slow-responding" devices, you may need to adjust these parameters upwards, often in combination of adjusting Lon Comm Config parameters (see "Lon Comm Config notes" on page 3-6).

Always In Zero Length Domain notes

By default, when commissioning or replacing a Lon device that has two domain entries (domain indexes 0 and 1), its second domain (domain index 1) is set to "not in use". In the case where the LonNetwork's working domain is *not* the zero-length domain, the device will *not* be a zero-length domain member.

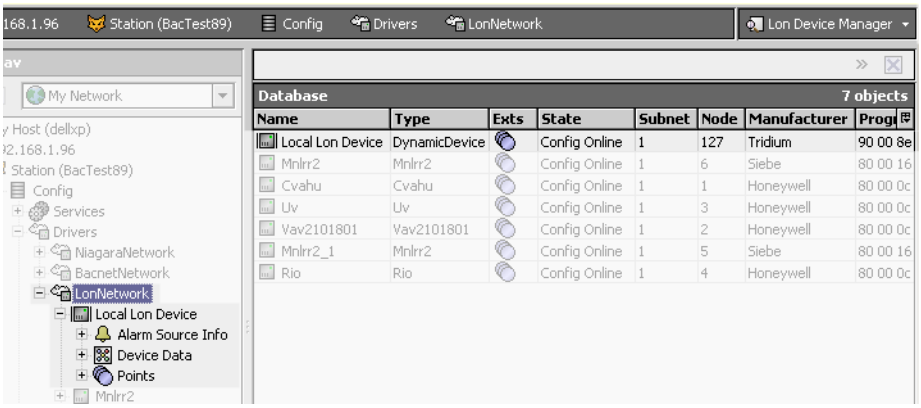
A property "Always In Zero Length Domain" is in Lon Netmgmt, with a default value of `false`. If left at default, a device's second domain entry is set to "not in use" as before. However, if set to true, a device with two domain entries will always be commissioned to be a member of the "zero-length" domain.

For related details, see "Domain Id notes" on page 3-9 and "Commission process" on page 3-15.

About the Local Lon Device

The Local Lon Device represents the local interface to the Lonworks network. Component-wise, it is modeled similar to any other Lon device. As a frozen slot under the LonNetwork, it always appears in the network (Figure 3-12). There is only one Local Lon Device—you cannot delete it or duplicate it.

Figure 3-12 *LocalLonDevice always in a LonNetwork*



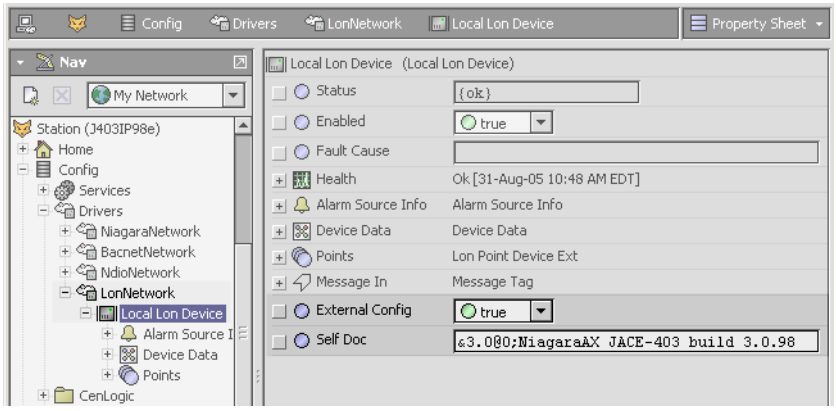
Typically, (with Station as network manager), you leave LocalLonDevice properties at defaults, and do nothing else at this level in the network. The following sections provide more details:

- LocalLonDevice External Config
- LocalLonDevice startup
- Local Nv Manager view
- LocalLonDevice actions

LocalLonDevice External Config

Two properties of the Local Lon Device are unique (not in any other Lon device), as shown in its property sheet (Figure 3-13). Values require change from defaults *only* if configuring the station as a Lon node only.

Figure 3-13 *LocalLonDevice properties for external Lonworks network management*



These two properties are:

- **External Config**
  By default, this is `false`. If set to `true`, this specifies that all Lonworks network management is performed externally, meaning that you do not use LonNetwork views like the Lon Device Manager, Lon Link Manager, and so on.
- **Self Doc**
  String for the JACE's (node) self documentation (if External Config is true), with a default value:
  `&3.0@;Niagara Server Node`
  Up to 1024 bytes are permitted. Note that a single asterisk (*) *omits* self documentation.

Note: *Whenever using self documentation, the leading header portion (`&3.0@`) should always be retained, along with an additional zero (`0`), plus other characters as described in the next section, "LNS device self-documentation notes". Also see "Notes when configuring as Lon node" on page 3-40 for related details.*

LNS device self-documentation notes For LNS compatibility, in the Local Lon Device's "Self Doc" property value, you must either:

- simply enter a single asterisk (*) to *omit* device self documentation, *or*
- enter a self documentation string that includes all LonMark objects used in the station, using the following format:

  `&3.0@0NodeObjectName,FunctionalBlock;selfDocText`

  where, from (left-to-right):
  - `&` character denotes an interoperable device.
  - `3.0` identifies the major and minor version of LonWorks Interoperablility Guidelines used.
  - `@` is used as a separator.
  - 0 is the first of the indices of the corresponding functional block index (0, 1, 2, etc.), where the Node object must be the first of the indices (0).
  - *`NodeObjectName`* is the description of the Node object.
  - *`FunctionalBlock`* includes any other LonMark or non-LonMark functional profiles that you wish to add to the station database.
  - *`SelfDocText`* is any text that you might want to add to describes the functional profiles.

For example:

```
&3.0@0NodeObject;NiagaraAX Server Node
```

Note: *For LNS compatibility, the Node object represented by the LocalLonDevice also requires two mandatory network variables:* `SNVT_obj_request` *and* `SNVT_obj_status`*. You must create these network variables using the LocalLonDevice's* **`Local NV Manager`** *view. See "Local Nvs required by LNS" on page 3-42 for details.*

LocalLonDevice startup

Upon first execution (if Neuron Id is 0), the following sequence occurs:

1. Address is set to default: channel Id 1, subnet node 1/127.
2. DeviceData is filled in from local neuron: Neuron Id, Address Count, Two Domains.

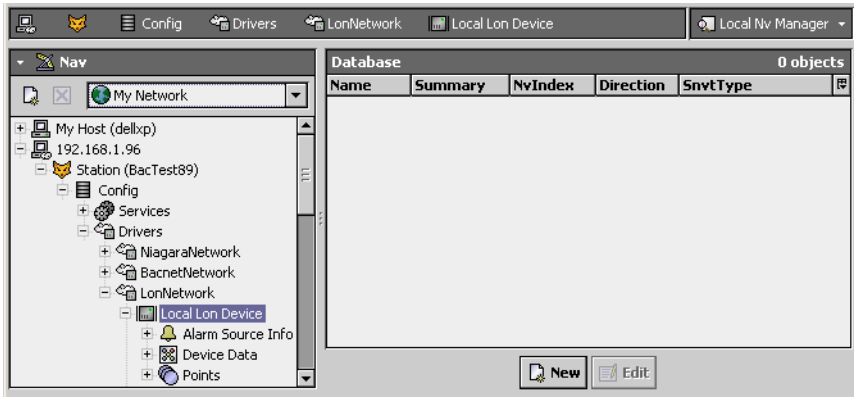
Upon station startup, the following occurs:

1. Updates to domain table, address table, and device state per station database.
2. Program Id is set.
3. Executes ping.

Local Nv Manager

As shown in Figure 3-14, the *default view* of the Local Lon Device is the **`Local NV Manager`**. This view is *unique* to the Local Lon Device (note that the standard Lon device views are *not* available).

Figure 3-14 *Local Nv Manager is default view for Local Lon Device*



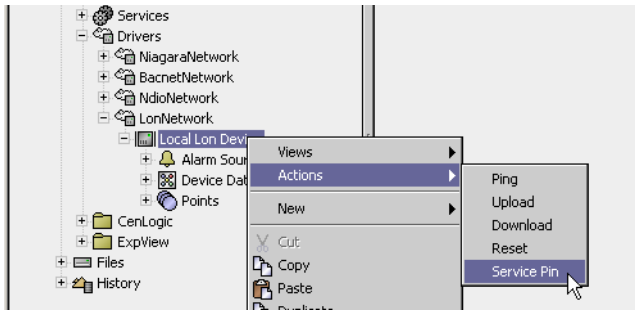
Note: *In typical operation, the station acts as the Lonworks network manager—you do not use this view (or methods) to share Niagara data among other Lon devices. Instead, you create Lon proxy points under Lon devices, as needed, and then link the proxy points into station logic.*

However, if configuring the station as only another "Lon node," this view lets you create "custom" Lonworks network variables (nvis, nvos, ncis) available externally to other Lonworks devices. The station appears as a "peer" Lonworks device. In this scenario, Lonworks network management is *not* handled by the station. For further details about configuring a station in this manner, see "Notes when configuring as Lon node" on page 3-40.

LocalLonDevice actions

By default, the Local Lon Device has five available *actions*, with right-click menu access from the LonNetwork wire sheet, LocalLonDevice property sheet, Nav side bar (see Figure 3-15).

Figure 3-15 *LocalLonDevice actions*



These are similar to other Lon device actions, and include the following:

- Ping
- Upload
- Download
- Reset
- Service Pin

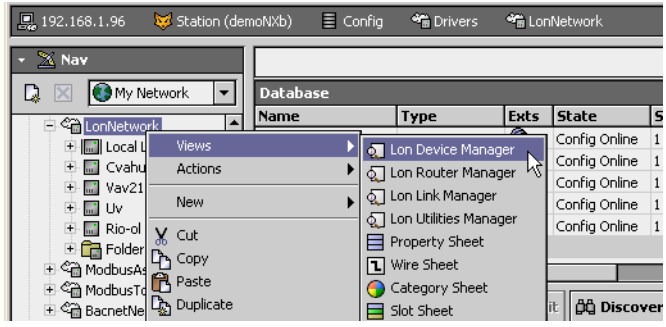
Service Pin The Service Pin action is unique to the LocalLonDevice, and is useful if installing the JACE using an external Lonworks network management tool (station as a Lon node only). This action issues a Lonworks service pin message for the JACE node.

Note: *In general, you should not invoke actions on the LocalLonDevice, except a Service Pin (as described in the scenario above).*

About Lon Network views

To facilitate network management, the LonNetwork provides four different manager views, as shown in Figure 3-16.

Figure 3-16 *LonNetwork views*



These manager views are as follows:

- **Lon Device Manager**
  This is the default view. See "About the Lon Device Manager" on page 3-14.
- **Lon Router Manager**
  See "About the Lon Router Manager" on page 3-18.
- **Lon Link Manager**
  Use this view to manage link types and bind links in the Lon network. See "About the Lon Link Manager" on page 3-20.

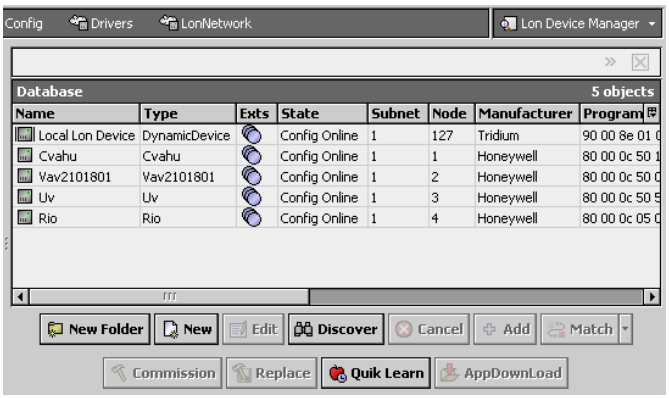
- **Lon Utilities Manager**
  Use this view to perform various utilities that are useful for managing and troubleshooting a Lon network. See "About the Lon Utilities Manager" on page 3-25.

Note: *In addition to the LonNetwork's four "manager" views, and its property sheet (Figure 3-7 on page 6), you also typically use the network's wire sheet view. This is where you can review and engineer Lonworks (peer-to-peer) bindings directly between Lon nodes, graphically pulling wires between Lon devices. See "About the LonNetwork wire sheet" on page 3-28.*

About the Lon Device Manager

The Lon Device Manager (Figure 3-17) is the *default view* for a LonNetwork—simply double-click the LonNetwork in the Nav side bar, or within the station's **`Driver Manager`** to see this view.

Figure 3-17 *Lon Device Manager (not Learn Mode)*



The Lon Device Manager provides support for learning (or discovering/adding) Lonworks devices to the database, for managing device addresses, and for downloading standard applications to devices.

Note: *An available "Path" column is also in the* **`Lon Device Manager`**, *which you can enable in the table options menu. Its intention is to allow for easier sorting of devices by parent LonDeviceFolder (reflected in path), a technique often used on a routed Lon network, where devices may be on different network channels (channel Id and subnet). For related details, see "About the Lon Router Manager" on page 3-18.*

The following sections provide more details:

- Lon Device Manager key points
- About Commission
- About Replace
- About Quik Learn
- About AppDownload

Lon Device Manager key points

In many ways, the Lon Device Manager works similar to other device managers that support online device discovery. See "About the Device Manager" in the *Drivers Guide* for general information.

However, the **`Lon Device Manager`** is *unique* from other managers in the following ways:

- A *second* row of buttons exists below ones common to most device managers. These buttons are only available when *not* in "learn mode" (split panes, "Discovered" and "Database").
  These buttons are:
  - **Commission**
    To set a selected device's internal tables (including address-related) to a functioning but unbound state. For details, see "About Commission" on page 3-15.
  - **Replace**
    To download network management data to a selected device, used when replacing a device, or when re-synchronizing nv bind information. For details, see "About Replace" on page 3-16.
  - **Quik Learn**
    To both discover and automatically add Lon devices in the station, representing new nodes found in the working domain. For details, see "About Quik Learn" on page 3-16.
  - **App Download**
    To download a vendor-supplied "nxe" file, such as may be used for a firmware update or application update. For details, see "About AppDownload" on page 3-17.
- Following a **`Discover`**, any Lon node already represented in the station's LonNetwork does *not ap-*

pear in learn mode's "Discovered" table (see Figure 3-18 on page 15). This differs from other drivers, where an existing discovered device appears "ghosted" in the "Discovered" table.

- The **Match** function has an available *toggle* in which you select whether to use the subnet-node address of the discovered (network) node, or have address management performed automatically by the Lon Device Manager—see Figure 3-19 on page 15.
  - Choose the first ("Use Net Subnet/Node") if working with a previously-managed network.
  - Choose the second ("Use Db Subnet/Node") if working with an unmanaged network.

Figure 3-18 *Lon Device Manager (Learn Mode following a Discover)*

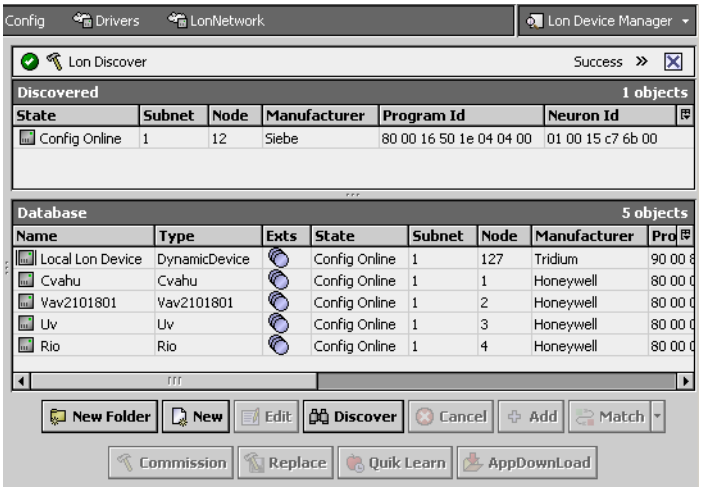
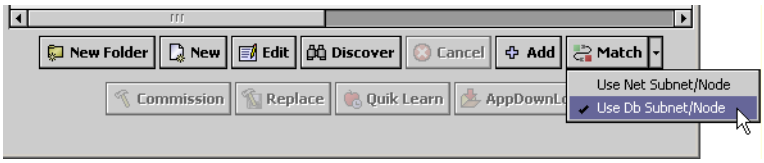


Figure 3-19 *Match function has address source toggle*



About Commission

Commission in the Lon Device Manager is used to set a device's internal tables to a functioning but unbound state. You typically perform this on any newly-added Lon device *except* those created by a **Quik Learn** of a previously managed network, where the initial state of the device is already "`Config Online`."

Note: *You can examine a device's/discovered node's internal tables using the Lon Utilities Manager.*

For more details, see Commission process.

Commission process A commission command results in the following steps:

Note: *Steps marked * do not apply if commissioning LocalLonDevice.*

1. * If using a service pin, the process waits for a service pin message to obtain the nodes Neuron ID, otherwise, the Neuron ID already stored is used.
2. * Needed "device data" is read from the device (for example: hosted, two domains, address count, address of snvt self-doc table).
3. The device's domain table is initialized, as follows:
   - Domain index 0 is set to the LonNetwork's working domain.
   - If the device has two domains, domain index 1 is set to "not in use".

   Note: *A property "Always In Zero Length Domain" is in the LonNetwork's "LonNetmgmt" component, with a* `false` *(default) value. If set to* `true`*, and the LonNetwork's working domain is* ***not*** *the zero-length domain, the second domain entry is set to the zero-length domain. See "Always In Zero Length Domain notes" on page 3-10.*

   - Use Netmgmt authentication key and devices subnet/node address in all active domains.
4. All entries in the address table are set to "not in use."
5. * All entries in nvconfig are set to unbound selector (0x3FFF - nvIndex). If under LonNetmgmt, VerifyNvDir is set, a check is made that device and database nv's have matching direction.
6. If there is a configSrv nv, it is set to external.

7. Update configuration device data as follows:
   - channelId: per property under LonNetMgmt.
   - nodePriority: per Lon device property.
   - location: per Lon device property.
   - authenticate: per property under LonNetMgmt.
8. Node is set to state `Configured, online`.

About Replace

The Replace function in the Lon Device Manager downloads network management config data to a selected device. As the name implies, you use it when physically replacing a device (you have removed the old device, and replaced it with an identical type device).

Note: *Even more typically, you use* **`Replace`** *on the same (original) device, in case you have run a verify report from the Lon Utility Manager, and saw errors reported for that device. Typically, after performing binds from the Lon Link Manager, it is a good practice to run such a verify report—this ensures each device's nv bind configuration matches the network management stored in Niagara. For more details, see "About the Lon Utilities Manager" on page 3-25.*

For more details about replace, see Replace process.

Replace process A Replace command results in the following steps:

1. If using a service pin, the process waits for a service pin message to obtain the nodes Neuron ID, otherwise, the Neuron ID already stored is used.
2. Verification is made that the new (replacement) device's program ID matches the programId stored in the existing Lon device.
3. The device's domain table is initialized, as follows:
   - Domain index 0 is set to the LonNetwork's working domain.
   - If the device has two domains, domain index 1 is set to "not in use".

   Note: *A property "Always In Zero Length Domain" is in the LonNetwork's "LonNetmgmt" component, with a* `false` *(default) value. If set to* `true`*, and the LonNetwork's working domain is* ***not*** *the zero-length domain, the second domain entry is set to the zero-length domain. See "Always In Zero Length Domain notes" on page 3-10.*
   - Use Netmgmt authentication key and devices subnet/node address in all active domains.
4. All entries in the address table are set to match the station database.
5. All entries in nvConfig are set to math the station database.
6. Update configuration device data as follows:
   - channelId: per property under LonNetMgmt.
   - nodePriority: per Lon device property.
   - location: per Lon device property.
   - authenticate: per property under LonNetMgmt.
7. Node is set to state `Configured, online`.

Note: *You can examine a device's/discovered node's internal tables using the Lon Utilities Manager.*

About Quik Learn

`Quik Learn` in the Lon Device Manager is a function that combines online node discovery and Lon device (database) creation in one operation. You often use it to populate a new LonNetwork with the proper device components. See "Using Quik Learn for a new LonNetwork" on page 2-3.

You can also use **`Quik Learn`** when adding new devices to an already configured LonNetwork (in this scenario, select the "Unconfigured network" option).

In addition, Quik Learn operation differs depending on if you have devices selected or not when you issue the command.

Note: **`Quik Learn`** *searches locally-installed* `lon<vendor>` *modules to find possible lnml files. This differs from a* **`Discover`** *and* **`Add`***, where all modules used by Workbench are searched. For the best possible results, make sure that any needed lon modules are installed on the JACE platform before running Quik Learn.*

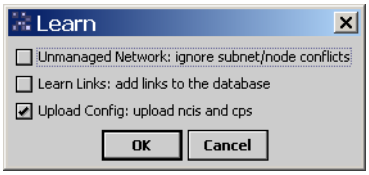
Note also that starting in *AX-3.3*, Quik Learn can optionally create "LonObject" containers for learned LonComponents. For further details, see *"Use Lon Objects notes" on page 3-10.*

For more details, see:

- Quik Learn, no device selected
- Quik Learn, device(s) selected

Quik Learn, no device selected If you click Quik Learn with no devices in the database selected, a popup dialog asks about previous network management, and whether to learn links, with config upload preselected, as shown in Figure 3-20.

Figure 3-20 *Learn dialog from Quik Learn, no device selected*

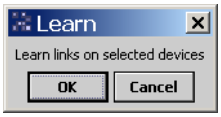


The following process occurs during the Lon Learn job.

1. A discovery occurs for all nodes on the *working domain* (only). As each device is discovered:
   - If you specified a managed network, then unconfigured nodes are ignored, and a check is made for duplicate subnet-node address conflicts. If conflicts are found, the learn is aborted.
   - Device data is retrieved for each node, including neuronId, programId, nodeState, authenticated, twoDomains, workingDomain, and channelId.
   - Attempt is made to match with existing device (subnet, node, programId with neuronId=0). If not matched, device data is saved for this node until all devices are discovered.
2. Remaining unmatched devices are processed. For each unmatched device:
   - Attempt is made to match with existing device (programId match with neuronId=0).
   - If no match possible, a new DynamicDevice is created, where the node's program ID is used to find the appropriate lnml file among the host's locally installed `lon<vendor>` modules.
3. As each device is matched or created, the database's device data is sync'ed with data read from the physical node.
4. If learn links was selected:
   - For each device, upload address, nvConfig, and alias tables.
   - For each device add links.
   - location: per Lon device property.
   - authenticate: per property under LonNetMgmt.
5. If upload config was selected, for each device upload values of its ncis and cps.

Quik Learn, device(s) selected If you click Quik Learn with one or more devices in the database selected, a popup dialog asks if you wish to learn links, as shown in Figure 3-21.

Figure 3-21 *Learn dialog from Quik Learn, device selected*



If you click **`Cancel`**, the learn is aborted. If you click **`OK`**, the following process occurs during the Learn Lon Links job:

1. For each device selected:
   - Subnet, node address, and domainId are verified. If not a match, then the learn is aborted.
   - Device data is retrieved, including programId, nodeState, authenticated, twoDomains, workingDomain, and channelId.
   - Uploaded are address, nvConfig, and alias tables.
2. For each device selected:
   - Add links.

About AppDownload

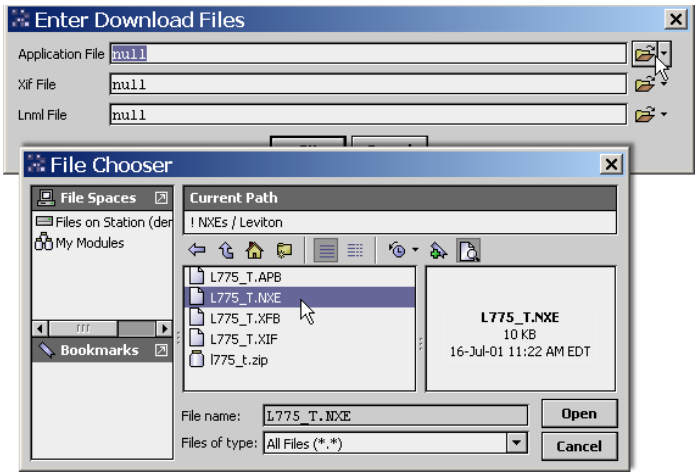
The **`AppDownload`** function in the Lon Device Manager provides a means to download vendor-supplied application file (of type: `*.nxe`) to one or more selected Lon devices.

Note: *An `*.nxe` file contains a binary application image for loading in a specific Lonworks device. If the download changes the "external interface" of the device, you must also supply either an `*.xif` or `*.lnml` file to correctly set the nv data, which is represented in station database separate from the device's application. Different external interfaces should be associated with unique program IDs. Some vendors support multiple device types in the same hardware platform.*

For more details, see AppDownload process.

AppDownload process When you select a device and click **AppDownload**, a popup dialog asks if you to specify the application (*.nxe) file. You click the folder icon (for the **File Chooser**), and then navigate to select the needed nxe file, as shown in Figure 3-22.

Figure 3-22 *Enter Download Files and File Chooser dialogs, from AppDownload*



If the application download will change the device's external interface, you should also select either a matching .xif or .lnml file (if you supply both, the .xif is used and the .lnml is ignored).

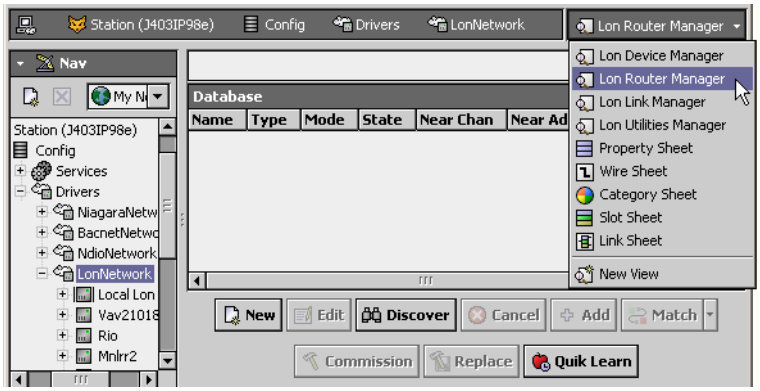
After you have selected the file(s) and click **OK**, the following occurs:

1. Verification is made that the supplied file(s) are for the same device type.
   If files are for a different programId than the device, a prompt is issued to verify that the change of device type is acceptable.
2. Issue a warning about the action about to be taken, asking for confirmation.

About the Lon Router Manager

The Lon Router Manager view of the LonNetwork (Figure 3-23) provides support for discovering and adding Lonworks routers to the database, and for managing router addresses.

Figure 3-23 *Lon Router Manager view of a LonNetwork*



Note: *If no Lonworks routers are installed on the physical network, you can safely ignore this view.*

The following sections provide more details:

- Rules for a routed Lonworks network
- Lon Router Manager key points
- About Router Commission

Rules for a routed Lonworks network

A routed Lonworks network configuration must follow certain rules:

1. A channel is a network segment between routers. A particular subnet must be contained in a single channel. There can be multiple subnets on a single channel.
2. The network must be configured such that there is only one path between and two specific nodes. There cannot be any loops.

3. Routers have two Neuron chips (and IDs). Each Neuron is a member of a separate channel, and must be assigned to a unique subnet/node address.

Note: *Routers have a "far" and "near" interface relative to the network management device.*

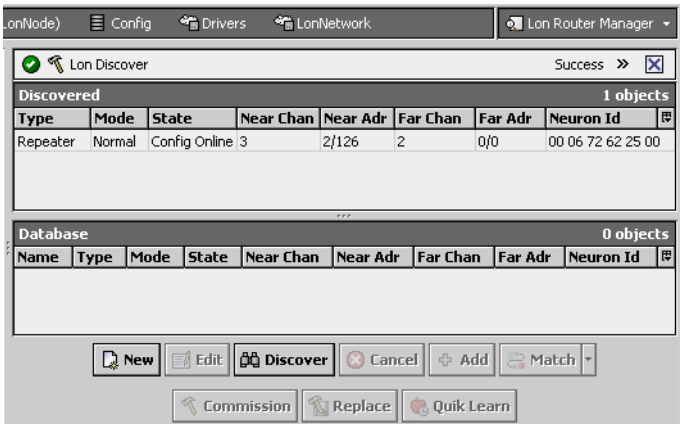
Niagara LonNetwork router rules When configuring a routed LonNetwork, you must:

1. Wire the routers according to the above rules.
2. Assign Channel Ids and Subnet Ids that match the wired configuration and these rules.

Lon Router Manager key points

In many ways, this view works like the **`Lon Device Manager`**. See "About the Lon Device Manager" on page 3-14 for more details. However, the Lon Router Manager is *unique* in that it is for management of Lon routers (only), meaning you do not see other (non-router) Lon devices in it.

Figure 3-24 *Example discovered Lonworks router*



As in the **`Lon Device Manager`**, there is a *second* row of buttons below ones common to most device managers. These buttons are available when *not* in "learn mode" (split panes, "Discovered" and "Database").

These buttons are:

- **Commission**
  To set a selected router's internal tables (including address-related) to a functioning state. For details, see "About Router Commission" on page 3-19.
- **Replace**
  Performs the same function on a router as a Commission.
- **Quik Learn**
  To both discover and automatically add routers in the station. Generally, this is recommended only for "previously managed" routed networks. This works much the same as in the Lon Device Manager. For general information, see "About Quik Learn" on page 3-16.

About Router Commission

Commission in the Lon Router Manager is used to set a router's internal tables to a functioning state. You typically perform this on any newly-added router *except* those created by a **`Quik Learn`** of a previously managed network, where the initial state of the router is already "`Config Online`."

Note: *You can examine a router's internal tables using the Lon Utilities Manager.*

For more details, see Router commission process.

Router commission process A commission command results in the following steps:

1. If using a service pin, the process waits for a service pin message to obtain the router's Neuron ID, otherwise, the Neuron ID already stored is used.
2. The router's domain table is initialized, both far and near side, as follows:
   - Domain index 0 is set to the LonNetwork's working domain.
   - If the device has two domains, domain index 1 is set to either:
     – "not in use" (if the working domain is the zero-length domain), or
     – zero-length domain (if the working domain is *not* the zero-length domain).
   - Use Netmgmt authentication key and devices subnet/node address in all active domains.
3. Update configuration device data as follows:
   - channelId: per property under LonNetMgmt.

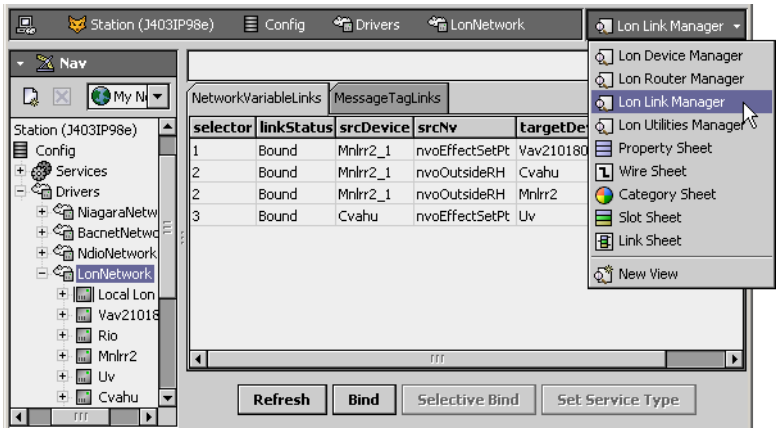
- nodePriority: per Lon device property.
- location: per Lon device property.
- authenticate: per property under LonNetMgmt.

4. The router table is set per the current network configuration.
5. Router is set to state `Configured, online`.

About the Lon Link Manager

The **`Lon Link Manager`** view of the LonNetwork (Figure 3-25) is where you can review/manage the bindings of network variables. As the Niagara station is (typically) "the network manager" of the Lonworks network, you often use this view.

Figure 3-25 *Lon Link Manager view of a LonNetwork*



See the following additional **`Lon Link Manager`** sections for more details:

- Lon Link Manager tabs
- Data columns
- Lon Link Manager commands

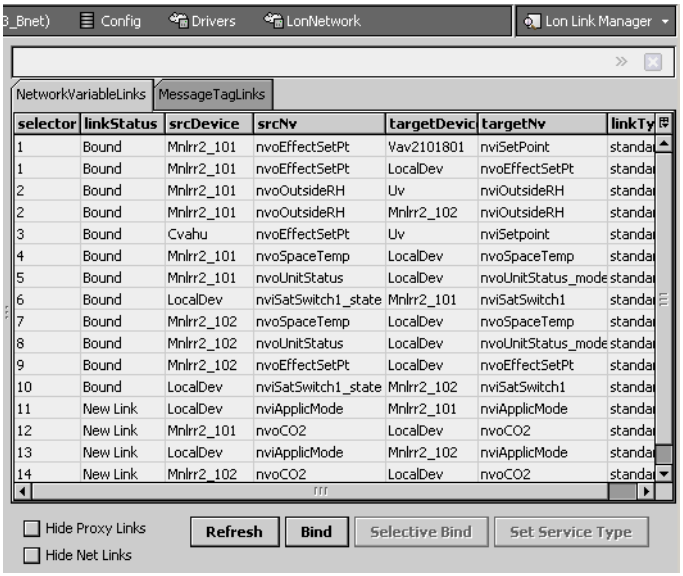
Lon Link Manager tabs

The Lon Link Manager has two separate tabs, where each provides a table-based view:

- NetworkVariableLinks — the default tab, used to review and manage links/bindings between network variables.
- MessageTagLinks — to manage bindings between "message in" and "message out" tags of Lon devices (if any devices are used this way).

NetworkVariableLinks NetworkVariableLinks is the default display tab in the Lon Link Manager. Often, this is the only tab you use in this view.

Figure 3-26 *NetworkVariableLinks tab in Lon Link Manager*



selector	linkStatus	srcDevice	srcNv	targetDevic	targetNv	linkTy
1	Bound	Mnlrr2_101	nvoEffectSetPt	Vav2101801	nviSetPoint	standa
1	Bound	Mnlrr2_101	nvoEffectSetPt	LocalDev	nvoEffectSetPt	standa
2	Bound	Mnlrr2_101	nvoOutsideRH	Uv	nviOutsideRH	standa
2	Bound	Mnlrr2_101	nvoOutsideRH	Mnlrr2_102	nviOutsideRH	standa
3	Bound	Cvahu	nvoEffectSetPt	Uv	nviSetpoint	standa
4	Bound	Mnlrr2_101	nvoSpaceTemp	LocalDev	nvoSpaceTemp	standa
5	Bound	Mnlrr2_101	nvoUnitStatus	LocalDev	nvoUnitStatus_mode	standa
6	Bound	LocalDev	nviSatSwitch1_state	Mnlrr2_101	nviSatSwitch1	standa
7	Bound	Mnlrr2_102	nvoSpaceTemp	LocalDev	nvoSpaceTemp	standa
8	Bound	Mnlrr2_102	nvoUnitStatus	LocalDev	nvoUnitStatus_mode	standa
9	Bound	Mnlrr2_102	nvoEffectSetPt	LocalDev	nvoEffectSetPt	standa
10	Bound	LocalDev	nviSatSwitch1_state	Mnlrr2_102	nviSatSwitch1	standa
11	New Link	LocalDev	nviApplicMode	Mnlrr2_101	nviApplicMode	standa
12	New Link	Mnlrr2_101	nvoCO2	LocalDev	nvoCO2	standa
13	New Link	LocalDev	nviApplicMode	Mnlrr2_102	nviApplicMode	standa
14	New Link	Mnlrr2_102	nvoCO2	LocalDev	nvoCO2	standa

As shown in Figure 3-26, each row in this view represents either:

- A connection between the station (*LocalDev*) and a specific nv (nvi or nvo) in a Lon device. Each rows corresponds to a particular Lon *proxy point*.
  Note: *Proxy points for ncis, cps, and "read-only" nvi proxy points are not included.*
- A Lonworks nv binding directly between two *other* Lon devices. On each end, the binding is modeled in the station as a "graphical" connection (link) visible on the devices' glyph (shape). See "About the LonNetwork wire sheet" on page 3-28.

Data columns provide the current information about each link/binding.

Note: *The NetworkVariableLinks tab of the Lon Link Manager provides two checkboxes in the lower left of the view (see Figure 3-26). These filter (hide) links from the view as follows:*

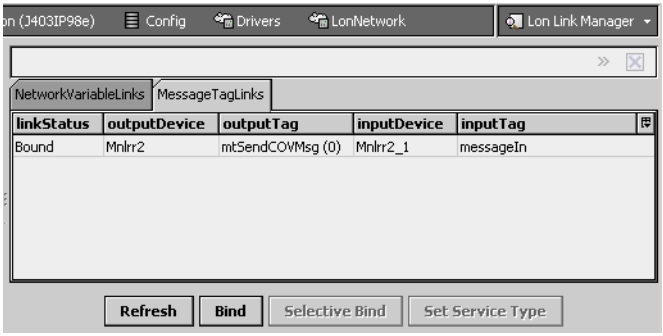
- Hide Proxy Links — Hides links from or to proxy points (LocalDev) from the view.
- Hide Net Links — Hides links between other Lon devices/controllers from the view.

Note that the hide filters only affects the display of links. A **`Bind`** *operation will still bind all links, even though they might not be displayed.*

MessageTagLinks The MessageTagLinks tab (Figure 3-25) is available in the Lon Link Manager, although not typically used. Generally, Lonworks nodes use network variables to exchange data.

Note: *In some cases, applications require a different data interpretation model than that provided by network variables. In these cases, nodes construct individual messages and assign them an address. These are referred to as explicit messages, and nodes can use logical input and output ports (message tags) to send and receive these messages.*

Figure 3-27 *MessageTagLinks tab in Lon Link Manager*



linkStatus	outputDevice	outputTag	inputDevice	inputTag
Bound	Mnlrr2	mtSendCOVMsg (0)	Mnlrr2_1	messageIn

All LonMark-certified nodes contain a "message-in" tag, which can be used to receive messages. In addition, nodes can declare bi-directional message tags that can be used to both send and receive messages. If message tags bindings are used, the Lon Link Manager displays their status in a fashion similar to that used to display network variable bindings.

Data columns in MessageTagLinks table provide the current information about each message tag link.

Data columns

Data columns in both Lon Link Manager tables are predefined, using different NetworkVariableLink columns and MessageTagLink columns. As needed, you can click on table column headers to resort the table, and click and drag to resize column widths.

NetworkVariableLink columns In the NetworkVariableLinks table, the following data columns appear, from left-to-right:

- **selector**
  Niagara selector number used to reference an individual network variable, stored in a table within the local Neuron chip. By default, the table is *sorted* by selector, ascending (1, 2, 3, so on).
- **linkStatus**
  Current status of each link, for example "Bound," "New Link," and so on. For details, see "Types of link status" on page 3-24.
- **srcDevice**
  Niagara name for the source Lon device, meaning *output side* of the link/binding. Any entry with "LocalDev" indicates a Lon proxy point for an *nvi* in the targetDevice.
- **srcNv**
  Niagara name of either:
  - LonComponent (nvo) under srcDevice, providing srcDevice is not LocalDev.
  - Lon proxy point for nvi in targetDevice, if srcDevice is LocalDev.
- **targetDevice**
  Niagara name for the target Lon device, meaning *input side* of the link/binding. Any entry with "LocalDev" indicates a Lon proxy point for an *nvo* in the srcDevice.
- **targetNv**
  Niagara name of either:
  - LonComponent (nvi) under targetDevice, providing targetDevice is not LocalDev.
  - Lon proxy point for nvo in srcDevice, if targetDevice is LocalDev.
- **linkType**
  Reflects the assigned Lonworks service type of the link (if bound). By default, new links use the "Standard" service type. You can selectively change the service type for any number of selected entries in this table, using the Set Service Type button. See Lon Link Manager commands for more details.

MessageTagLink columns In the MessageTagLinks table, the following data columns appear, from left-to-right:

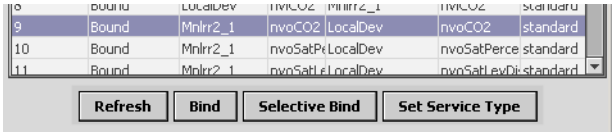
- **linkStatus**
  Current status of each message link, for example "Bound," "New Link," and so on. For more details, see "Types of link status" on page 3-24.
- **outputDevice**
  Niagara name for the source Lon device, meaning device with the *message-out tag*.
- **outputTag**
  Niagara name for the message tag in outputDevice.
- **inputDevice**
  Niagara name for the target Lon device, meaning device with the *message-in tag*.
- **inputTag**
  Niagara name for the message tag in inputDevice.

Lon Link Manager commands

Note: *Two commands are "global," whereas the other two (Selective Bind, Set Service Type) require you to select (click) entries in the table. Note that source nvs using structured data are not individually selectable—if you select one element in a nv's structure, any other element entries (e.g., proxy points for the same nv) are also automatically selected.*

At the bottom of the Lon Link Manager view (Figure 3-28) are four command buttons.

Figure 3-28 *Lon Link Manager commands (buttons)*



Commands Refresh, Bind, Selective Bind, and Set Service Type are described as follows:

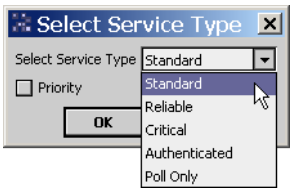
Refresh Globally polls the status of the bindings on the network and displays that information in the table. See "Types of link status" on page 3-24. If unbound (deleted) links were previously listed, those entries are removed from the table.

Bind A "global command" to establish nv bindings for *all* entries, per the "linkType" (service type) specified for each entry. The **`Bind`** button remains available regardless if you any items selected. When you click it, a global "Lon Bind job" is performed, with standard station job visibility. To bind only *selected* items, use **`Selective Bind`** instead.

Selective Bind Selective Bind is enabled only when you have one or more items selected in the table. Use it to establish nv bindings only for those selected entries, per the "linkType" (service type) specified in each entry. When you click it, a "Lon Bind" job is performed, with standard station Job visibility. To globally bind all entries, use the **`Bind`** button instead.

Set Service Type Set Service Type is enabled only when you have one or more items selected in the NetworkVariableLinks table. This allows you to set the LonTalk message service type, controlling parameters used for the next bind. When you click it, a popup dialog (Figure 3-29) provides a drop-down selection of message service types to use. Or, you can select "Poll Only"—this sets nvs to an *unbound* state.

Figure 3-29 *Select Service Type dialog*



Selections "Standard, Reliable, Critical, Authenticated" set parameters per settings in the LonNetwork's LonNetmgmt "Link Descriptors," where timers are applied to the address table entry used for the link. Other parameters are applied to the Nv ConfigData on nvs (output side of the link).

Also, a **`Priority`** checkbox is available. If you enable (check) this in combination with any of the message service types, upon a bind, the Priority flag is set in the nv's Nv Config Data.

Assigned message service type displays in the far-right "linkType" column in the Lon Link Manager. See "NetworkVariableLink columns" on page 3-22.

Note: *If a link is already bound and you change it to another message service type, its linkStatus changes to "NewLink" until you* **`Bind`** *(or* **`Selective Bind`***) it again.*

About message service types The four LonTalk message service types available for a nv binding can described as follows:

- **Standard**
  (unacknowledged) is the least reliable, yet often most widely-used. This is the default service type for a Niagara binding. With this service, the message is sent one time and no response is expected. This service is typically used when the highest level of network performance is needed, network bandwidth is limited, and the application is not sensitive to the loss of messages.
- **Reliable**
  (unacknowledged/repeated) is recommended for messaging that has been classified as important—if there is a specific need to know that a status has changed. Reliable messaging sends a message three times. The srcDevice sends each message three times, and if the targetDevice receives the message, it simply throws subsequent messages away.
- **Critical**
  (acknowledged) means that the targetDevice must acknowledge the fact that it received a message. The srcDevice continues to send the message until the targetDevice acknowledges receipt of that message. You are limited to the number of bindings that can be defined as critical, because it taxes the system, and often causes communications issues (by loading down the network with message verification traffic).
- **Authenticated**
  is a security service reserved for secure messaging. When using authentication, the srcDevice must identify itself to the targetDevice, and vice-versa. This service is rarely used, and only in instances where messaging must be secure.

Types of link status

Each link/binding in the Lon Link Manager appears with a status in the second column, "linkStatus." Typically, link status is one of the following values (assuming no error status):

- **NewLink**
  Appears for either of the following:
  - Proxy point was added, yet **Bind** command (or **Selective Bind** for it) has not been given.
  - A link was created between two nodes, yet the **Bind** command (or **Selective Bind** for it) has not been given.

  Until bound, entries do not exist in the address tables of the nodes, and polling is used exclusively. Following a **Bind** or **Selective Bind** command, link status should change to "Bound." For more details on these commands, see "Lon Link Manager commands" on page 3-22.
- **Bound**
  Reflects a Lonworks binding established between two network variables/nodes. If Quik Learn was used to learn links in an existing (managed) network, learned links will appear as bound. In addition, following a Bind command, all entries for Lon *proxy points* will also list as bound.
- **Poll Only**
  Appears whenever you have Set Service Type to "Poll Only." Upon the next **Bind** command, nvs are set to unbound state.
- **Obsolete**
  Appears for either of the following:
  - Any proxy point (that was formerly bound) that was deleted.
  - A bound link between two nodes that was deleted.

  In either case, the address tables in both nodes have not yet been updated—entries still exist. Use the **Bind** command to change these Obsolete entries to *Unbound*.
- **Unbound**
  Appears for obsolete links following a **Bind** command, just to verify that the link has been deleted. To remove these entries (rows) entirely, click the **Refresh** command button.

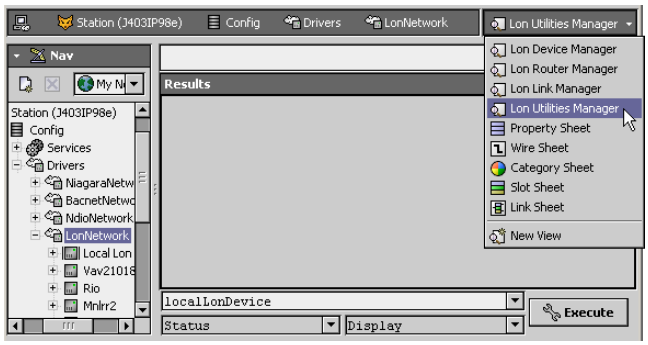
Error status In some cases, a link/binding may appear in the Lon Link Manager showing an error-type linkStatus, as one of the following types:

- AliasError — One or more connections have been made which require the use of alias nvs when none or insufficient aliases are available. Delete links and bind to remove error.
- AuthicateError — Authentication error. Check the messaging service type for the network variable linkage.
- ComError — Communications error occurred during the bind operation (for example, timeouts). Try binding the network variables again. If errors persist, check communications.
- DeviceError — Device error. The state of the node will not support a network variable binding.
- DirtyGroup — Typically means an address table problem. Usually, a second pass at a bind will clear things up.
- DirtyPoll — An existing group has been disrupted by commissioning or removal of one of its member devices. Bind to reassign groups.
- Error — An error apart from one of the other specified error types.
- GroupError — Unable to assign the node to a group. A node can belong to 15 different groups (up to 15 address table entries) when acknowledged messaging service is used. To correct, either reduce the number of address table entries, or adopt unacknowledged/repeated messaging service.
- GroupExcludeError — A set of connections has been created for which no set of groups can be assigned that meet all group assignment rules. Delete links and bind to remove error.
- NvTypeError — SNVT types do not match. Either the Lon xml (lnml) file is not appropriate for the node, or (if Learn Nv was used) the node's self-documentation does not properly represent the actual data stored in that device. Obtain an up-to-date lnml file for the device, or use Learn Nv.
- MaxCritError — Too many targets (bindings or linkages) are made to use the critical messaging service. Reduce the number of bindings or adopt a non-critical messaging service.
- PriorityError — The priority setting of the link cannot be legally accommodated (that is, priority is selected but nv is not priority, and priority is not configurable).
- ServiceTypeError — Message service type mismatch. The assigned message service type (linkType) does not match the actual service type found in the node.

About the Lon Utilities Manager

The **Lon Utilities Manager** view of a LonNetwork (Figure 3-30) provides the means to perform a variety of queries to both devices in the LonNetwork and discovered nodes.

Figure 3-30 *Lon Utilities Manager view of a LonNetwork*



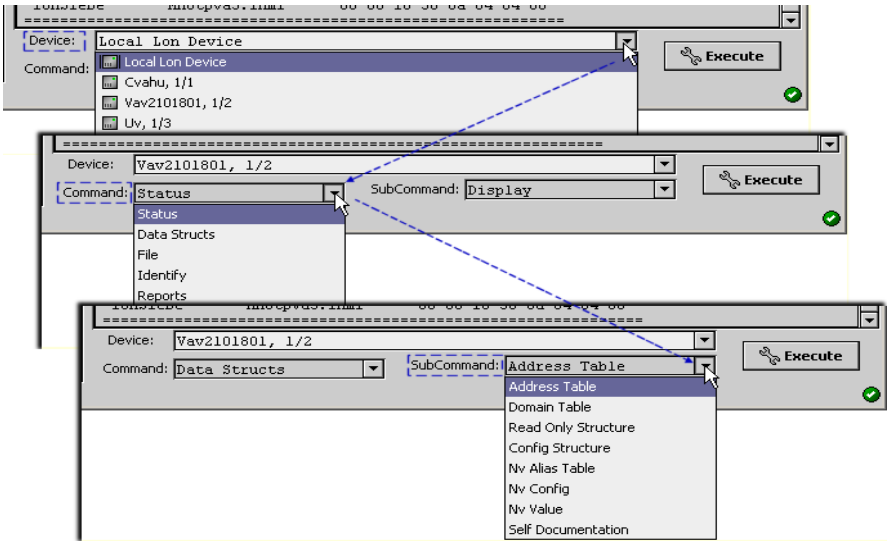
See the following additional **Lon Utilities Manager** sections for more details:

- Building a utility command
- Types of Lon utility commands
- Common usages of Lon Utilities Manager

Building a utility command

Build a command in the Lon Utilities Manager by using drop-down controls at the bottom of the view.

Figure 3-31 *Building a command in Lon Utilities Manager*



As shown in Figure 3-31, to build a command you:

- Select *Device*—list includes both devices in the database, as well as discovered nodes.
- Select main *Command*—list includes Status, Data Structs, File, Identify, Reports, Find.
- Select *SubCommand*—list differs according to the selected command ("Data Structs" sub-commands are shown in Figure 3-31).

After you build the command, click **Execute** to see results in the utilities manager view.

Note: *Starting in AX-3.5 and build 3.4.55 and later, text labels appear beside the selection fields for Device, Command, and SubCommand, as shown in Figure 3-31.*

Types of Lon utility commands

Main commands available in the Lon Utilities Manager include the following:

Status, Data Structs, File, Identify, Reports, Read Mem, and Find.

Status Use this command to display real-time status for a node, or to perform some other network management operation that affects its status. The following sub-commands are available:

- **Display**
  Issues a query status message to the node. This retrieves the network error statistic accumulators, cause of the last reset, current state of node, and the last runtime error logged.
- **Clear**
  This sends a message to the node to clear its error flag and error counts. After this is processed, the status display for the node updates. Following a clear, you can use the display sub-command to gauge the effects of network traffic on the node.
- **Set Unconfigured**
  This sets the node's internal and database state to unconfigured, and displays new node status. In an unconfigured state, a node's application is loaded, but configuration data is either not loaded or deemed to be corrupted due to a configuration checksum error.
  When unconfigured, a node responds to status query messages with its Neuron ID rather than subnet/node addressing. Network variables do not update, and its service LED blinks at a once-per-second rate.
- **Set Online**
  This sets a node to an online state, and is provided as a convenience for a device not represented in the station database. From a Lon device's property sheet, under the Device Data slot, you can directly access and write to a device's Node State. See "DeviceData notes" on page 3-30.
- **Reset**
  Issues a software reset to the node, and displays new status.

Data Structs Use this command to display various internal device tables and structures. The following sub-commands are available:

- **Address Table**
  Defines the network addresses to which the node can send implicit messages and nv updates, and also the groups to which the node belongs. Up to 15 address entries are possible.
- **Domain Table**
  Defines the domain(s) to which the node belongs. Up to two domains can be assigned.
- **Read Only Structure**
  Defines the node's identification as well as some of the application image properties of the node.
- **Config Structure**
  Defines the hardware and transceiver properties of the node, and resides in its EEPROM. Some portions are written by the manufacturer, while other fields are written when the node is installed.
- **Nv Alias Table**
  Defines configuration attributes of the alias network variables in the node. Aliases allow an nv in one controller to be linked to multiple nvs in another (single) controller.
- **Nv Config**
  Defines the configurable attributes of the network variables in the node. Up to 62 entries are possible.
- **Nv Value**
  Displays raw (hex data) values for nvs in the node, without any conversion (as provided for the nv LonComponents or nv proxy points). Can be used to verify against values seen in Niagara.
- **Self Documentation**
  Displays the self-documentation available in the node. This capability must be provided for a device to have the LonMark logo.

File This command lets you display internal files of any selected Lonworks device, providing that it supports files. The following sub-commands are available:

- **File Directory**
  Shows list of available files, including type and size.
- **Config Template File**
  Shows template file entries.
- **Config Value File**
  Shows config value file entries, and bytes.
- **Other**
  Shows files other than template and value files.

Identify Use these commands to identify a node in the field. Basically, commands are either "wink" (send *to* node) or "service pin" (send *from* node). The following sub-commands are available:

- **Wink**
  Use this to send a *wink message* to a selected node, whereby, depending on the device, it typically visually (or audibly) indicates receiving the message—for example, flashing an LED in a pattern.

Note: *Before executing this command, select the specific device from the drop-down list.*

- **Service Pin**
  (Device selection is not used for this command.) This command causes the Lon Utilities Manager to listen on the network for a node to identify itself. In this mode, the utilities manager displays "`Waiting on a service pin`."
  When the service pin of a Lonworks device is pushed, that node sends its domain table to the utilities manager, where it displays in the view. For more details, see "Service pin notes" on page 3-10.
- **Clear Service Pin**
  Use this command to cancel a pending Service Pin command.

Reports Use these commands to display various reports for networked nodes, including network management data, transmit errors data, and to discover (verify) inconsistencies in the station database (against what is actually stored inside nodes that are online).

The following sub-commands are available:

- **Netmgmt Summary**
  Provides a summary of network links, address table entries, and group assignments.
- **Program Ids**
  Lists the current program IDs with associated module and lnml file or class.
- **Transmit Errors**
  Creates and displays a table of various transmission error counts for all nodes on the network. This command results in a "status clear" sent to all nodes, such that error counts are reset—the next report shows counts since this report was created.
- **Transmit Errors No Clear**
  Creates and displays a table of various transmission error counts for all nodes on the network. Does not clear device status in nodes (unlike **`Transmit Errors`** command).
- **Verify**
  Compares the networked devices' nv configuration and address table entries against the station's database (Lon device's properties) and reports discrepancies.
  Depending on the selected device, the verify report scope varies:
  - If *LocalLonDevice is selected* device, the report *verifies all* networked Lon devices.
  - If any *other device is selected* device, the report verifies *only that device.*
- **Network Summary**
  Creates a table of devices showing channel, subnet/node address, and Neuron ID. Also shown are routers (if any).

Read Mem This command provides a (starting) **`address`** field and **`len`**(gth) field in which you can enter values, using hexadecimal notation, to read "raw" memory contents of the selected device. Results display in hex bytes as well as ASCII characters, using 16 byte rows listed by address location.

For example, if you enter address of `01FC` and len of `FF`, you see 16 rows of data from `01FC` through `02EC`.

Find This command discovers nodes on the network (that are not already represented in the station by a Lon device). It is equivalent to the **`Discover`** command in the Lon Device Manager.

Note: *There are no* **`Find`** *sub-commands, and selected device is not important.*

Common usages of Lon Utilities Manager

The following notes list commands that are commonly run from the Lon Utilities Manager.

- To identify a device from a service pin message:
  Main command: Identify, sub-command: **`Service Pin`**.
  For a procedure, see "To identify a device using its service pin message" on page 2-1.
- To determine the working domain of an existing managed network:
  Select <*a Lon device*>, main command: Reports, sub-command: **`Domain Table`**.
  For a procedure, see "To see a node's current domain table" on page 2-3.
- To *verify* that binds issued from the Lon Link Manager executed correctly, meaning that the actual devices and the station database are both synchronized:
  - If selective binds made to just *one* Lon device:
    Select <*that device*>, main command: Reports, sub-command: **`Verify`**.
  - If binds were made to *multiple Lon devices*:
    Select *localLonDevice*, main command: Reports, sub-command: **`Verify`**.

  If the report shows discrepancies, take the appropriate actions from the Lon Device Manager:
  - For any *device* shown in error, select it, then do a Replace.
  - If the *database* shows errors, select the affected device(s), then do a **`Quik Learn`**. See "Quik Learn, device(s) selected" on page 3-17.

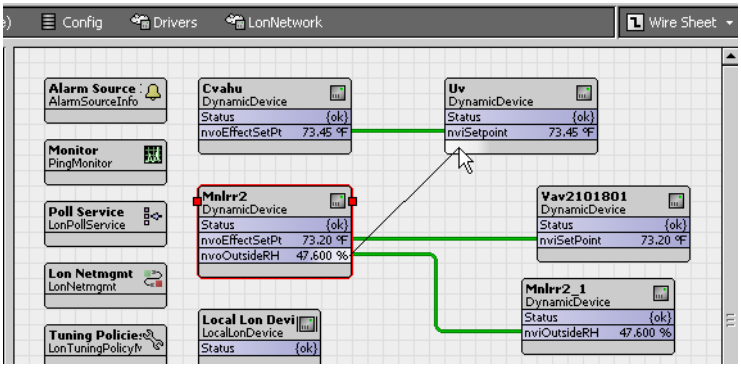
Following either action, you typically perform another Bind from the Lon Link Manager, then re-run the verify report from the Lon Utilities Manager.

About the LonNetwork wire sheet

Although not a "manager" view, whenever the station is configured as the Lonworks network manager (typical), the wire sheet view of the LonNetwork (and any child LonDeviceFolder containing Lon devices) is an *important* view. Here, bindings between Lon devices appear as graphical links (wires), and you can add or delete bindings between devices as needed. Figure 3-32 shows a link being pulled from an nvo of one device to another device, using the wire sheet of the LonNetwork.

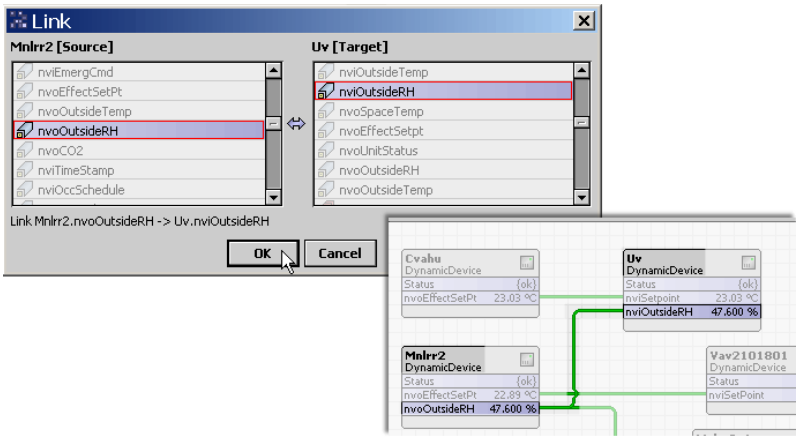
Note: *Using the wire sheet is not a strict requirement. For example, you can also perform the same operations using right-click commands (on Lon devices) in the Nav tree, such as "Link Mark" and "Link from <LonDeviceName>". However, the wire sheet graphically shows you existing bindings as wires or "nubs."*

Figure 3-32 *Adding link from LonNetwork wire sheet*



However you specify a link between Lon devices, the popup Link editor (Figure 3-33) provides nv (e.g. SNVT) verification to allow only legitimate links. When you click **OK**, a "new link" is added. See the next section, "New link to binding notes".

Figure 3-33 *Link editor allows only legitimate binds between devices*



New link to binding notes

When you graphically add a new link between Lon devices, say in the LonNetwork's wire sheet (Figure 3-32) and resulting Link editor (Figure 3-33), at this point it is still a "station dependent" link. In other words, until you perform a "Bind" or "Selective Bind" operation from the Lon Link Manager, the station is required for data exchange. See "Types of link status" on page 3-24.

As this "new link" state is counter to the Lonworks principle of "peer-to-peer" bindings (via nv updates), and is also inefficient in comparison, after adding links between devices you *should* bind them, using the Lon Link Manager. There, you can also set service types on bindings, if needed.

Note: *In the Lon Link Manager, a link/binding between two Lon devices appears as an entry row without "LocalDev" as the value of either "srcDevice" or "targetDevice" (whereas, any entry row with a "LocalDev" value represents a Lon proxy point in the station).*

About Lon devices

Lon device components are similar to most driver's device components— see "Common device components" in the *Drivers Guide* for general information. Among the standard slots are device status properties (including a Health container) and a "Device Alarm Source Info" container slot.

Each Lon device includes a **`Points`** extension for Lon proxy points. However, unlike a few other drivers, *other device extensions* such as **`Schedules`**, **`Histories`**, and **`Alarms`** are not used. Instead, other components unique to a Lon device appear.

The following sections provide more Lon device details:

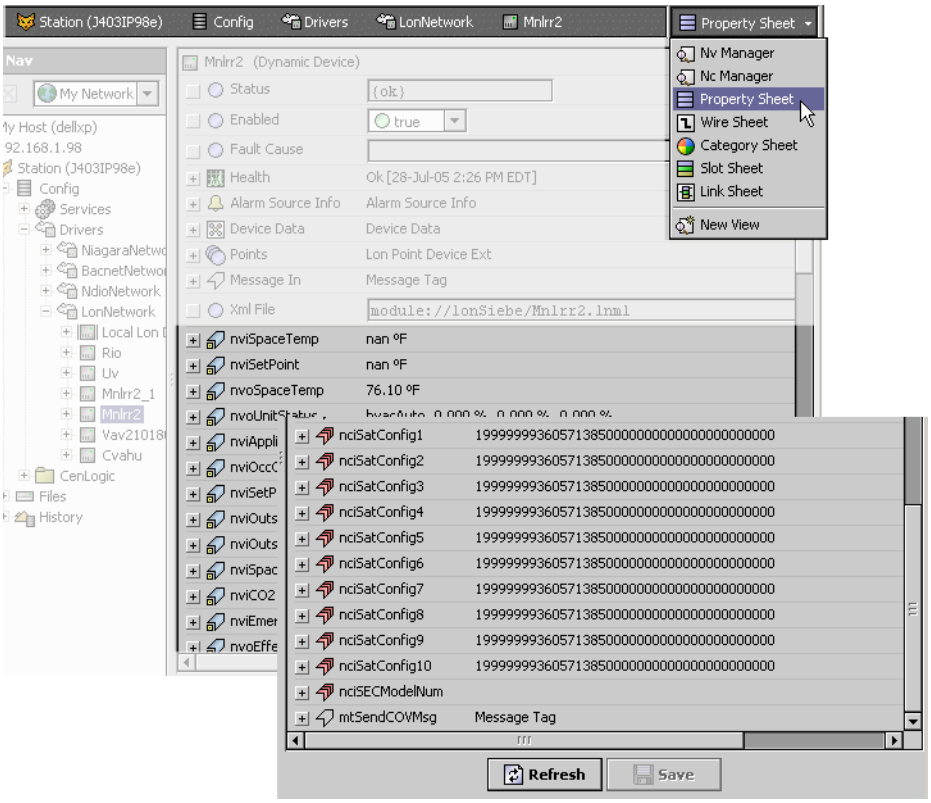
- Lon device overview
- ImportXml command
- DeviceData notes
- Lon device views
- Lon device actions

Lon device overview

In addition to standard device properties, each Lon device includes a "DeviceData" container, with a number of properties for status and configuration of that particular Lon device. Typically, you do not need to edit anything under this container. See "DeviceData notes" on page 3-30 for more details.

Also unique to a Lon device is its set of "LonComponents" that represent specific nvs (nvis and nvos), ncis, and cps for that device. A shown in Figure 3-34, main access to LonComponents is via the *property sheet* of the Lon *device*, where LonComponents are listed in "nv index" order below the "Xml file" property.

Figure 3-34 *LonComponent (nvs, ncis, cps) access from LonDevice property sheet*



For more details, see "About LonComponents" on page 3-35.

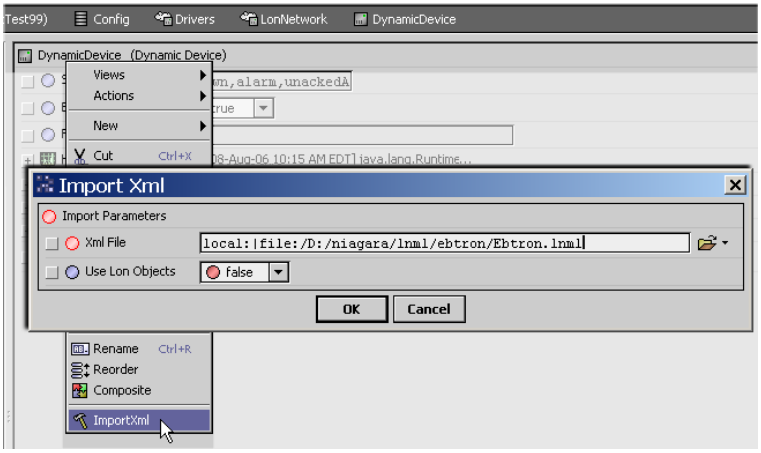
Also unique to a Lon device is a special Workbench command, special manager views, and right-click actions. See "ImportXml command" on page 3-30, "Lon device views" on page 3-31 and "Lon device actions" on page 3-33.

ImportXml command

Each Lon device (DynamicDevice) has an available right-click **ImportXml** command in Workbench. This command provides a popup dialog that displays the specified .lnml file—as stored in the device's "Lon Xml" property, and allows you to change it (if needed). See Figure 3-35.

Note: *Typically, this command is used only if programming offline, when devices (and their associated lnml files) cannot be automatically learned. This same functionality is automatically provided when you "Add" discovered Lon devices, providing that a Lon Xml file was found for the device.*

Figure 3-35 *ImportXml command for Lon device in Workbench (dialog shown is AX-3.3).*



Note: *Starting in AX-3.3, if the "Use Lon Objects" option in the* **ImportXml** *dialog is set to* `true`*, this results in an extra "folder" organization of a device's LonComponents, grouped by LonMark objects. For more details, see "LonObjects" on page 3-36.*

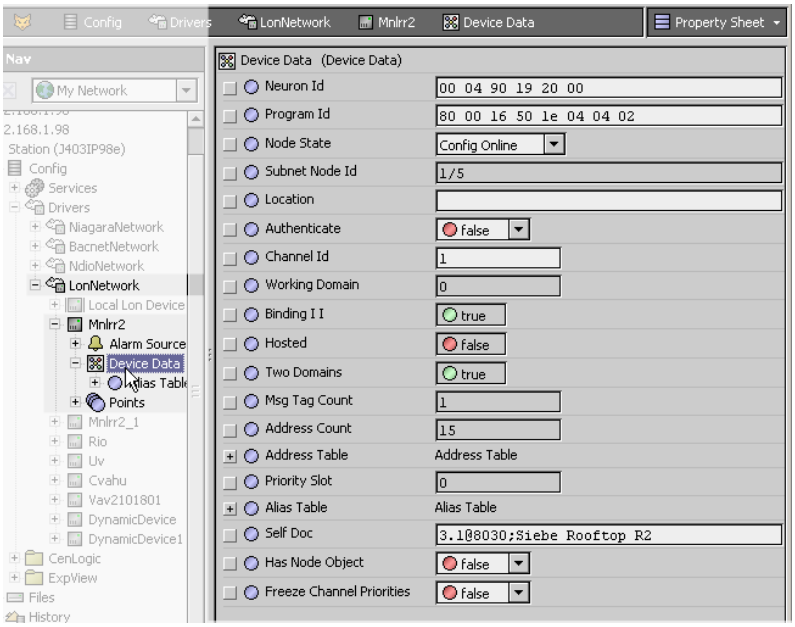
When you click OK to execute this command, Workbench reads in the specified `.lnml` file, and builds the various child LonComponents (nvs, ncis, cps) under the device.

Note: *If needed, you can build you own lnml files for devices, using the* **Lon Xml Tool** *in Workbench. See "Lon Xml Tool" on page A-1 for more details. After building an lnml file, you can specify it when adding Lon devices, or for devices already added use the command to rebuild its LonComponents.*

DeviceData notes

DeviceData appears in the Nav side bar under any Lon device, as shown in Figure 3-36. You can simply double-click it to see its property sheet.

Figure 3-36 *DeviceData is a container slot under any Lon device*



In general, it is recommended that you leave slot values under a device's **DeviceData** container at default settings. Largely, these are data structures that contain address and state information administered by the network management functions of the station.

Caution *Do not set the* **Node State** *for any "hosted" Lon node (or the Local Lon Device) to "Applicationless," or else the device will become irretrievably non-functional. Instead, use the Lon Device Manager view of the LonNetwork to make changes to Lon devices.*

One exception that may be routinely edited is the device's Neuron Id—this maps the device in the station database to a different physical device (or no device, if the address is not available).

Lon device views

By default, any Lon device (typically **DynamicDevice**) has two special *manager views*, in addition to the standard component views (property sheet, wire sheet, category sheet, slot sheet, and link sheet). These views are the **Nv Manager** and the **Nc Manager**.

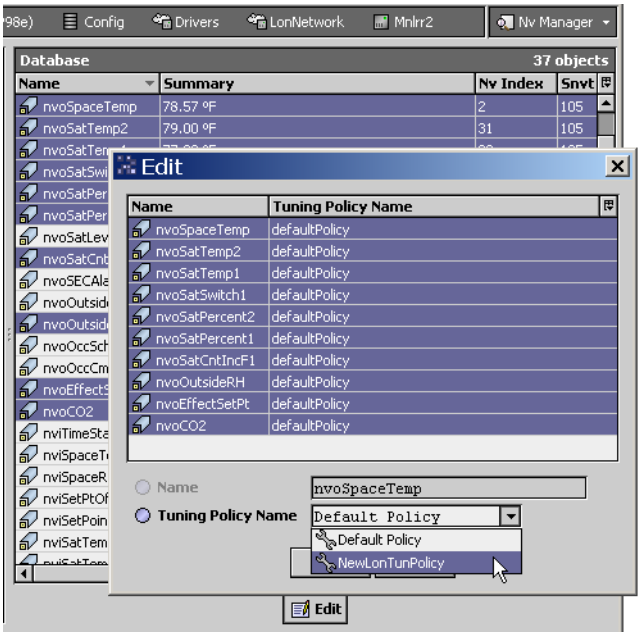
- About the Nv Manager
- About the Nc Manager

About the Nv Manager

The **Nv Manager** is the *default view* of any Lon device (except Local Lon Device). It allows you to browse through a table of the device's nvs (nvi and nvo) LonComponents, showing a number of sortable columns of data. You can use the standard table edit controls to change the number and types of data displayed in this view. Summary data is updated dynamically through polling.

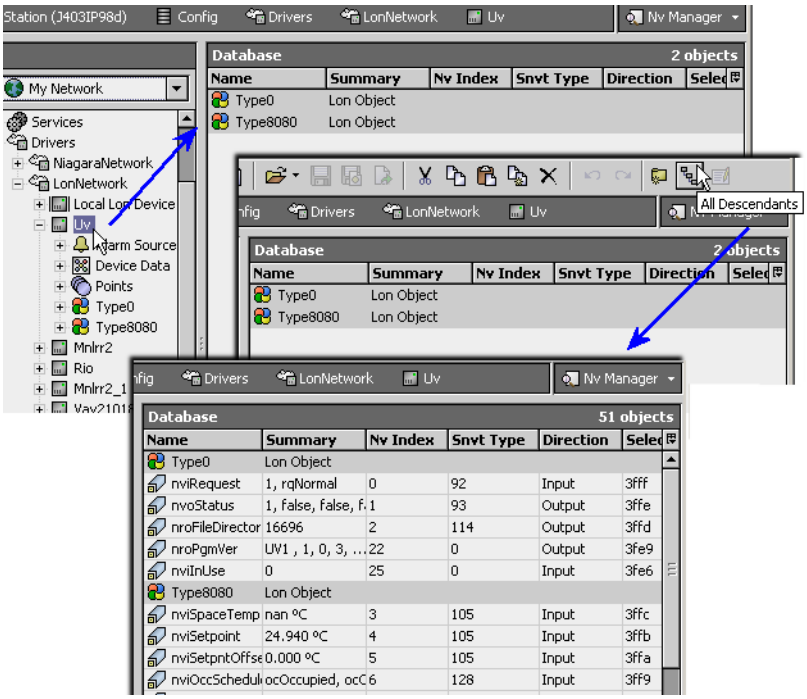
Edit access is also provided, where you can simply double-click any single row in the Nv Manager to change the components *name*, if needed. More typically, you use it to "gang edit" the *tuning policy* used by some group of nvs, as shown in Figure 3-37.

Figure 3-37 *Using Nv Manager and "gang edit" to reassign tuning policies to selected nvs*



Note: *If using the "Use Lon Objects" feature available starting in AX-3.3, the Nv Manager (or Nc Manager) view initially shows only the top-level LonObject(s), where each contains a set of LonComponents. In this case, click the "All Descendants" tool to see the nvs or ncis under each object, as shown in Figure 3-38.*

Figure 3-38 *Use the "All Descendants" tool in Nv Manager or Nc Manager view of device with LonObjects*

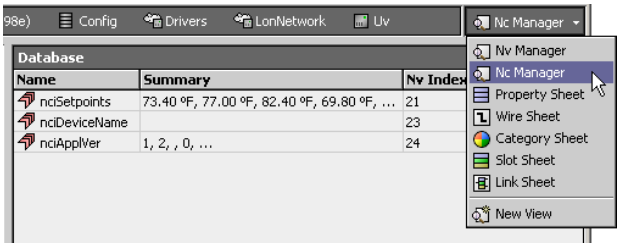


For details on globally setting the "Use Lon Objects" option, see "Use Lon Objects notes" on page 3-10. For details about LonObjects, see "LonObjects" on page 3-36.

About the Nc Manager

The **`Nc Manager`** is another available view of any Lon device. As shown in Figure 3-39, it is a table view similar to the Nv Manager, but provides access only to *ncis* in the device.

Figure 3-39 *NC Manager view on Lon device shows only ncis*



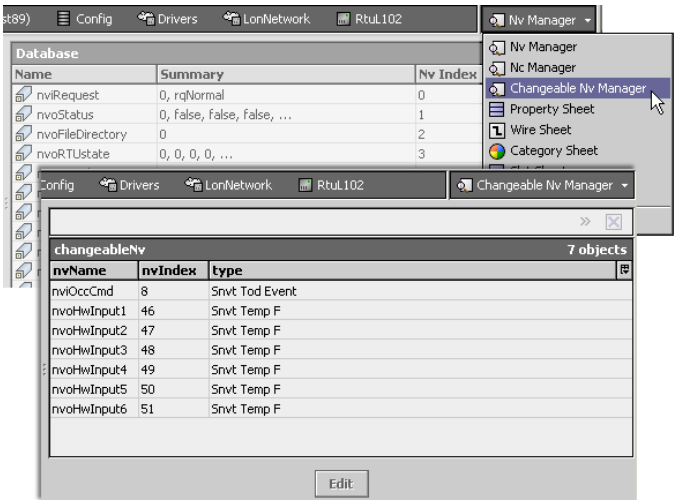
Again, table columns are resortable, and you can change the data columns for display using the standard table controls. Edit access to an nci using the Nc Manager is limited to changing the name of the associated LonComponent.

If LonObjects appear in the Nc Manager, click the "All Descendants" tool to flatten the hierarchy in the view to see ncis in each LonObject. See Figure 3-38 for an example doing this from the Nv Manager.

About the Changeable Nv Manager

Providing that a Lonworks device has the capability of "changeable-type network variables," the Lon device component for it has another available view: the **`Changeable Nv Manager`** (Figure 3-39).

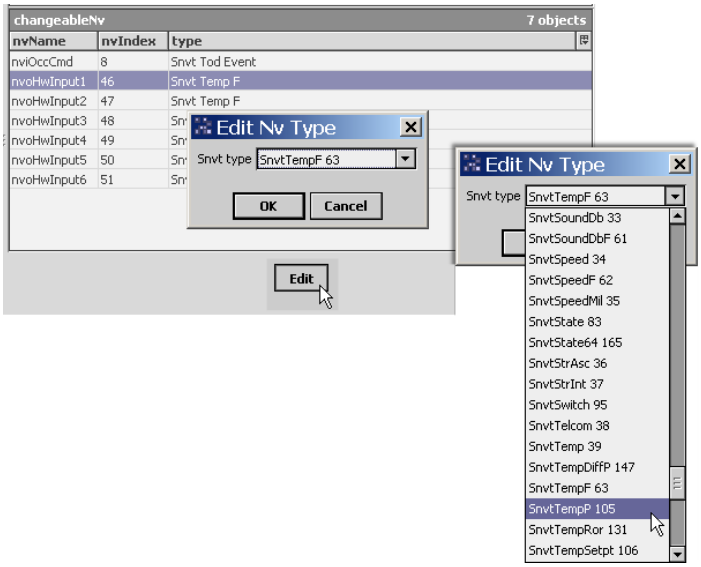
Figure 3-40 *Changeable Nv Manager view of a Lon device*



This table-based view lists all nvs in the device that are designated as changeable types, along with the current SNVT type used for each.

If needed, you can select one or more nvs and click the **`Edit`** button for an **`Edit Nv Type`** dialog, as shown in Figure 3-41.

Figure 3-41 *Editing selected nv in Changeable Nv Manager*



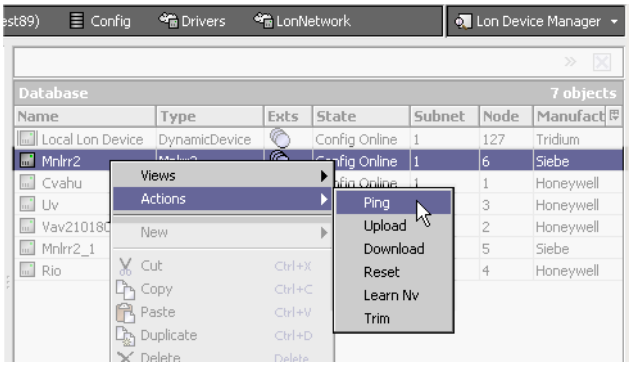
You can then select *another* SNVT type from the drop-down list (ordered alphabetically, each type also includes the SNVT type index *number*.) When you click **`OK`**, the nv(s) are changed in that device to use the new SNVT type.

Note: *If you select a SNVT type that requires more memory than allocated for that nv by the device, a "Max Length Error" popup appears, and the nv(s) remain unchanged.*

Lon device actions

By default, each Lon device has available *actions*, with right-click menu access from the Nav side bar, LonNetwork wire sheet, Lon device property sheet, or Lon Device Manager (see Figure 3-42).

Figure 3-42 *Lon device actions*



Lon device actions include the following:

- Ping
- Upload
- Download
- Reset
- Learn Nv
- Trim

Note: *Similar actions exist for the Local Lon Device, with exception of "Learn Nv" and "Trim." For more details, see "LocalLonDevice actions" on page 3-13.*

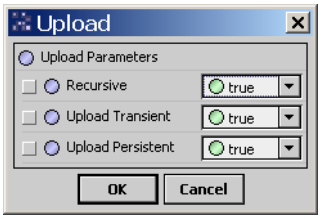
Ping

A Ping action attempts communication with device. If successful, the device status is ok. If this fails, the device status is set to down.

Upload

Upload reads transient (nvs) and persistent (ncis and cps) data from the device and writes to the station database (Lon device). An Upload dialog box (Figure 3-43) allows you to select the type of data.

Figure 3-43 *Upload Lon device dialog*



Typically, you leave dialog parameters at defaults (true)—recursive is always recommended. After using **`Discover`** and **`Add`** to create a new Lon device (and assigning the device a Lon Xml file), you can use **`Upload`** to populate current LonData in the device's LonComponents (nvs, ncis, cps).

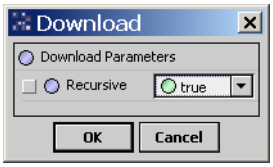
Note: *If using Quik Learn, an "Upload Config" option is already included, as shown in Figure 3-20 on page 17. This uploads all current LonData into the device's LonComponents, making another upload unnecessary.*

Note: *An **`Upload`** action is also available at the LonNetwork level—with the same Upload dialog selections as shown in Figure 3-43. This provides a "global upload" from all Lon devices.*

Download

Download writes persistent data (ncis and cps) to the device from values in station database (Lon device). A Download dialog box (Figure 3-44) allows you to select recursive writes.

Figure 3-44 *Download Lon device dialog*



Typically, you leave recursive at default (true), to write to all child data items. Use Download to restore nci and cp values to "known good" values, as previously saved in the station.

Note: *An* **`Download`** *action is also available at the LonNetwork level—with the same Download dialog selection as shown in Figure 3-44. This provides a "global download" to all Lon devices.*

Reset

A Reset action issues a reset command to the device.

Learn Nv

Learn Nv builds the Lon device's child *LonComponents* as a collection of nvs, ncis, and (if available) cps based on the Lonworks self documentation that resides in the device. See "About LonComponents" on page 3-35 for more details. As such, the Lon device will be represented with nvs that are implemented with SNVTs, and ncis and cps that are implemented with SCPTs.

Use Learn Nv *only* if there is no available Lon Xml (lnml) file for a device, or if the device has a programmable (changeable) external interface.

Note: *Be aware that if you perform a Learn Nv on any Lon device previously using a Lon Xml file, LonComponent data from any manufacturer-defined type info is removed (lost). Workbench does not allow a Learn Nv if the Lon device has an "Xml File" property that is not "null." Instead, a popup error message appears, saying that you must clear the* Xml File *entry and retry.*

Starting in AX-3.3, a Learn Nv can group the device's LonComponents in one or more LonObject containers under the Lon device, by selecting the "Use Lon Objects" option in the Learn Nv popup dialog (Figure 3-45). See "LonObjects" on page 3-36 for more details.

Figure 3-45 *Learn Nv popup dialog*

Note: *If you perform a Quik Learn and associated lon<Vendor> modules are not installed on that JACE platform (to supply lnml files), the "Learn Nv routine" is automatically used to build DynamicDevices, as needed.*

Trim

Trim can be used to automatically *remove* a Lon device's LonComponent nvi and nvo slots that do *not* have an associated Lon proxy point, or that are not linked or have a Px bind. This can conserve station memory and allows database support for more devices. See "About LonComponents" on page 3-35 for details about a device's LonComponents.

Note that the Trim action is simply a convenience—alternately, you can manually delete individual nvi/nvo LonComponent slots that have no interest. Typically, you would use Trim (or manually delete) only *after* adding all Lon proxy points and making any LonComponent binds, perhaps before duplicating the Lon device in multiples.

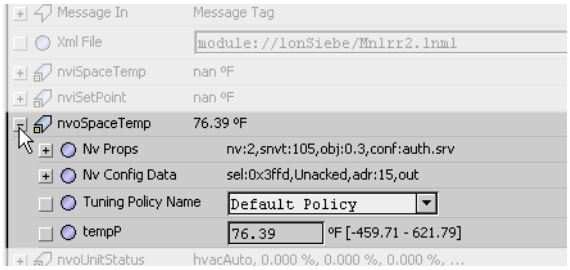
Note: *If you perform a Trim, and then later find you need access to deleted nvis and nvos, you can perform an ImportXml command on the device to add back deleted LonComponents.*

About LonComponents

Under any Lon device, all of its available nvs (nvis and nvos), ncis, and cps (SCPTs) are represented in the station by LonComponents, viewable on the device's property sheet (see Figure 3-34 on page 29). In turn, each LonComponent contains "LonData" plus or one or more container slots, including a "Props" structure needed by the station's network management to access or create links to that data.

As shown in Figure 3-46, you can expand any LonComponent in the property sheet of the device.

Figure 3-46 *Expand a LonComponent to see "Props", LonData, and (if nv) "Config Data"*



Note: *A Lon device's collection of LonComponents is determined by its referenced Lon xml (.lnml) file. The .lnml file is automatically determined in a "Quik Learn" (see "About Quik Learn" on page 3-16) or following a Discover in the Lon Device Manager. Or, you can manually specify the .lnml file in the* **Add** *or* **Edit** *dialog for a Lon device, then use the right-click device command "*`Import Xml`*".*

In the case of a DynamicDevice that does not have a referenced lnml, its network variables can be "learned" from a Learn Nv action (when the device is online with the station).

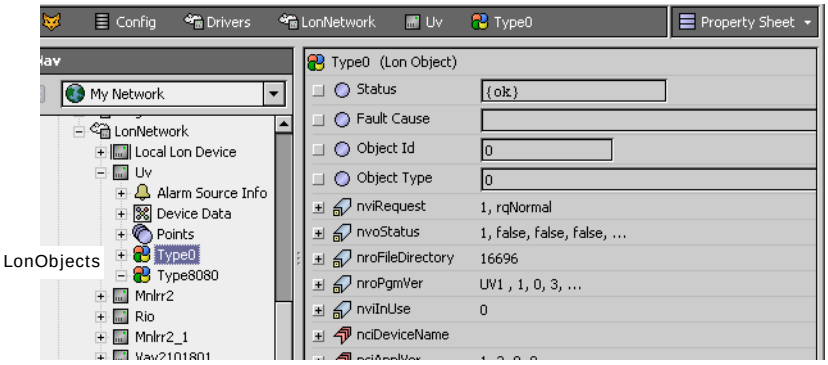
See the following sections for more details:

- LonObjects
- LonComponent slots
- Using LonData directly

LonObjects

Starting in AX-3.3, when learning/adding a Lon device, you can optionally specify for the device's LonComponents to be grouped under "LonObject" containers. Each LonObject represents a LonMark object, and its child LonComponents are the set of nvs, ncis, and cps that comprise that object.

Figure 3-47 *LonObjects group a device's LonComponents by LonMark object type*



As shown in Figure 3-47, LonObjects are visible in the Nav tree when the Lon device is expanded. This extra level of component hierarchy may be useful in devices that have a large number of LonMark objects.

The LonObject container has only a few properties, with important properties as follows:

- **Object Id**
  The numerical LonMark object identifier for the object, unique in the device among all objects.
- **Object Type**
  The numerical LonMark object type for the object, for example "0" for the "Node object," or "8080" for a "Unit Ventilator Controller" functional profile.

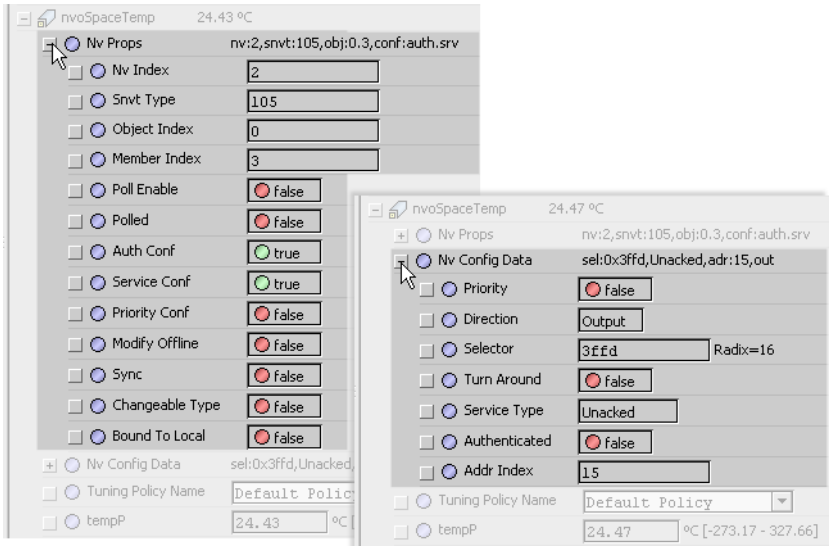
The default name for a LonObject is "`Typen`", where "n" is the LonMark object type number. The property sheet is the default view for a LonObject, as seen in Figure 3-47. For details on globally implementing LonObjects, see "Use Lon Objects notes" on page 3-10.

You can also specify the "Use Lon Objects" option on a Lon device that already exists in the station database, when issuing either the ImportXml command or the Learn Nv action.

LonComponent slots

For any of the LonComponents that represents an nv (nvi, nvo, or nci), in addition to a "NvProps" or "NcProps" slot it also has an "NvConfig Data" slot that maps directly to that device's network variable. Each "NvProps", "NcProps", and "NvConfigData" slot contains read-only properties, which you can expand and review if desired (Figure 3-48).

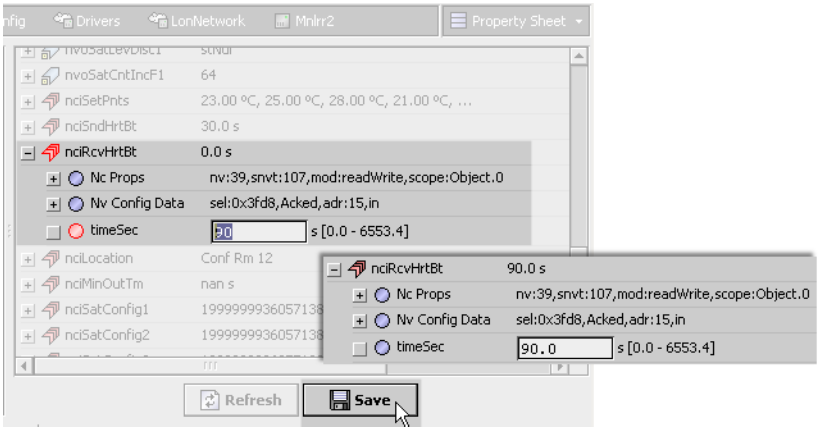
Figure 3-48 *Expanded "Nv Props", "Nv Config Data" for example nvoSpaceTemp*



Using LonData directly

With an online Lon device, you can access/use LonData values directly without making proxy points. For example, from the device's property sheet, you can expand any of the nci or cp LonComponents and review values, and if necessary make a value change. When you execute a **`Save`**, the change is written down to the device (Figure 3-49).

Figure 3-49 *Changing nci LonData in LonComponent*



You can also change nvi LonData values in the same way, however, nvi values are typically overwritten again. Note that you cannot directly change nvo LonData values (these are read-only), but for any LonComponent you can issue a "Force Read" action—see "LonComponent actions" on page 3-38.

Without requiring Lon proxy points, you can also make Px view usage of LonData.

Px view usage of LonData

For "simple monitoring" from Px views, you can "data bind" Px widgets to LonComponent ords to retrieve LonData values, without requiring proxy point creation. However, note the following about this LonData usage:

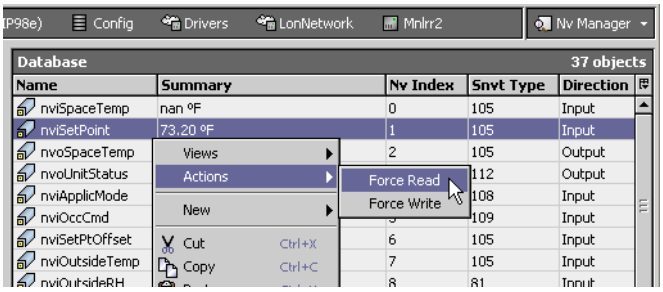
- Polling is used to fetch LonData values, whereas if you create proxy points for items you can (and typically do) create Lon binds to the LocalDevice (using the Lon Link Manager) to receive values via "nv updates." Overuse of polling may adversely affect Lon network traffic.
- Data logging or alarming (on value) requires a proxy point, where you add and configure history and/or alarm extensions, as needed.
- Using LonData values in station control schemes also requires proxy points.
- An nvi, nci, or cp represented with writable proxy points offer a Px view user "right-click" command access to set/override values. However, if Px view access to these items is via LonComponents (LonData), available actions are limited to "Force Read" (poll) or "Force Write" (resend value in station).

See the next section, "LonComponent actions".

LonComponent actions

By default, LonComponents have two available *actions*, with right-click menu access from the Lon device's property sheet, Nc Manager, or Nv Manager (see Figure 3-50).

Figure 3-50 *LonComponent actions*



The two actions are as follows:

- Force Read
- Force Write

Force Read

Force Read causes a read of the item's LonData from the device.

Force Write

Force Write re-sends the LonComponent's LonData values to the device (does not apply to nvos).

About Lon proxy points

Lon proxy points are similar to other driver's proxy points. See "About proxy points" in the *Drivers Guide* for more details. In addition to typical ProxyExt properties, the proxy extension in Lon proxy points include these other properties:

- **Target Comp**
  The target LonComponent represented by the proxy point, for example "nvoUnitStatus." See "About LonComponents" on page 3-35 for more details.
- **Target Name**
  The name of the LonData element in that LonComponent, for example "heatOutputPrimary".
- **Link Type**
  Link type used, as either Unknown, Standard, Reliable, Critical, Authenticated, or Poll Only.
- **Priority**
  Either true of false (default).

The following sections provide more Lon proxy point details:

- Lon Point Manager tips
- Lon proxy point type selection
- About Lon proxy point updates
- Misuse of Lon proxy points

Note: *For any U.S. installation, when creating Lon proxy points it is recommended that you change the parent point's (metric or SI) Facets to English equivalents, whenever appropriate. For more details, see "Facet conversion in Lon proxy points" on page 3-4.*

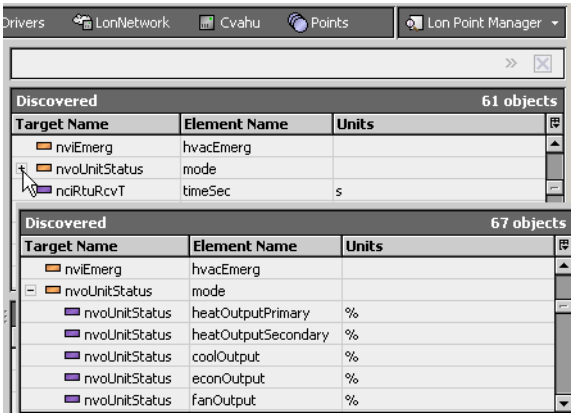
Lon Point Manager tips

In the **`Lon Point Manager`**, all available point data is already discovered when the device was initially learned. Or, if you added a DynamicDevice offline, and then referenced a specific Lon Xml file, the available point data is then known.

When you Discover (or toggle Learn Mode on) to see the discovered data (top pane), all data is listed by the names of the LonComponents (nvi*Name1* and nvo*Name2*, for example).

Many nvs and ncis are implemented with *structured* SNVTs—in this case (by default), you only see the *first* data element in the structure, and a "+" to the left of the nv or nci name.

Figure 3-51 *Structured nvs and nvis are expandable in Discovered pane*



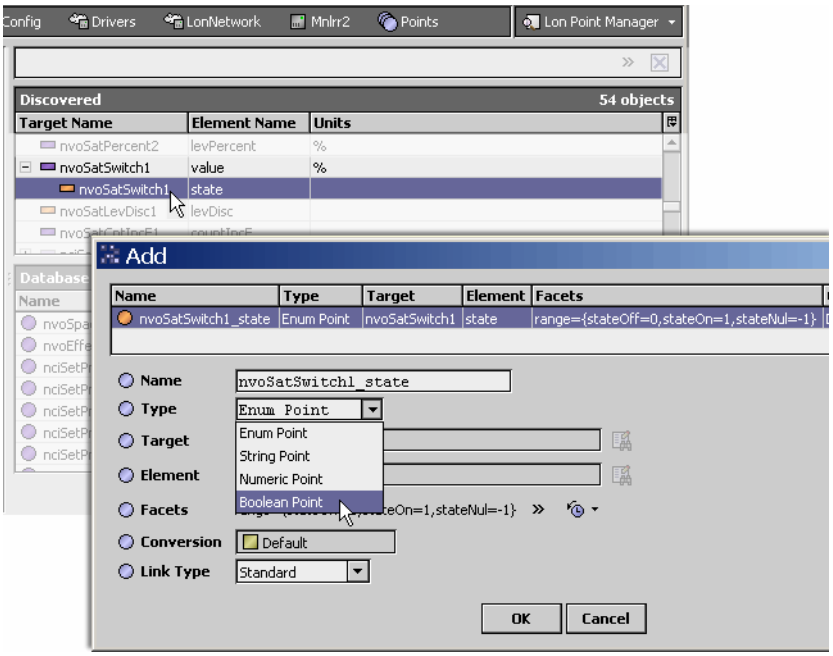
As shown in Figure 3-51, click the "+" to expand to see all data items in the SNVT structure. Each item is a potential candidate for a Lon proxy point. See the next section, "Lon proxy point type selection".

Lon proxy point type selection

When you add a Lon proxy point, a default Niagara point type is pre-selected. Often, the data type (category) is either numeric or enum, and less often, boolean or string. A Lon proxy point *supports default conversion* between data types. This means when *creating* a proxy point, you can select another data type, if useful in your station's application.

For example, if adding a proxy point for an nvo implemented with `SNVT_Switch`, and you are interested only in the "state" element as either false (Off) or true (On), and do not anticipate a "null" value, you can select a BooleanPoint instead of the pre-selected EnumPoint (Figure 3-52).

Figure 3-52 *Changing point type when adding Lon proxy point*



This would allow you to link the Out of the proxy BooleanPoint directly into a Logic-type kitControl object, without using an intermediate "conversion" object (StatusEnumToStatusBoolean). The proxy point would convert enum ordinals "0" to false, and "1" to true.

See Table 3-3 for how default Lon proxy point conversions are handled between data types.

Table 3-3 *LonData element type to proxy point data type, default conversions*

LonData element type	Proxy point type	Conversion method	Setup notes
Numeric	Boolean	False = 0, True = 1 value <=0 = False, >0 = True	—
	String	Convert between string representation and numeric values	—
	Enum	Numeric is enum ordinal	Add facet EnumRange
Boolean	String	True = "true", False = "false"	—
	Numeric	False = 0, True = 1 value <=0 = False, >0 = True	—
Enum	Numeric	Numeric is enum ordinal	—
	Boolean	True to ordinal 1, false to ordinal 0.	—
	String	Read/Write enum tag	—
String	Numeric	Convert between string representation and numeric values	—
	Enum	String is enum tag	Add facet EnumRange
	Boolean	True = "true", False = "false"	—

Also, when you add a proxy point under a Lon device for an nvi, nci, or cp, the **Add** dialog defaults to a *writable* type, for example NumericWritable or EnumWritable. Before creating the proxy point, you may wish to change to a read-only point type (if you wish to monitor only).

About Lon proxy point updates

By default, when you add Lon proxy points for nvs (nvi or nvo), each one is *polled* to obtain value updates. See "Network variable (nv) poll/update rules" on page 3-7.

Proxy points for ncis or cps (SCPTs) are never polled. These point values are obtained when you Upload from the Lon device (see "Lon device actions" on page 3-33), or issue **Force Read** actions to the corresponding LonComponents (see "LonComponent actions" on page 3-38).

Typically, instead of polling device's nvs for Lon proxy points (especially nvos), you *bind* those nvs to the LocalLonDevice. Then, updates occur using the "native Lonworks nv update" mechanisms, reducing the polling queue. You do these binds using the Lon Link Manager view of the LonNetwork. See "Bind Lon proxy points" on page 2-8 for procedures.

Misuse of Lon proxy points

Lon proxy points provide station access to nvs, ncis, and cps for usage in control, display, and various alarm and history schemes (via point extensions). However, do *not* link them to Lon proxy points in *other Lon devices on the same network* for the purposes of sharing nv data—this is a proxy point misapplication! Instead, create links directly between nvs of Lon devices, and then bind them in the Lon Link Manager. For related details, see "About the LonNetwork wire sheet" on page 3-28.

Notes when configuring as Lon node

If configuring the station as a Lonworks node (only), you do not use the various LonNetwork views that provide the interface to Lonworks network management. Instead, you work only in (and under) the Local Lon Device when engineering the station.

Note: *Before doing other configuration, set the* **External Config** *property of the LocalLonDevice from* `false` *(the default) to* `true`*, and make sure that its* **Self Doc** *property begins with the string "*`&3.0@0`*". See "LocalLonDevice External Config" on page 3-11.*

Following this, use the Local Nv Manager (default) view of the LocalLonDevice. As described in the following sections, you use a two-stage process to expose station data as Lonworks, where you:

- Add Local nvs and ncis
- Add Local Lon Device proxy points

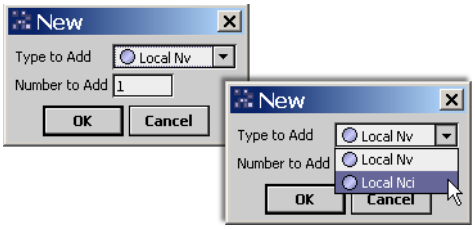
Note: *Starting in AX-3.4, you can also create an .xif file (external interface file) for the LocalLonDevice, typically done after completing the steps listed above. For details, see "LocalLonDevice XIF generation (AX-3.4 and later)" on page 3-45.*

Add Local nvs and ncis

Use the **Local Nv Manager** to define nvs and ncis to be available to other networked Lon nodes. Do this using the **New** button at the bottom of the manager, which produces the **New** dialog (Figure 3-53). In this dialog, you specify how many to add (and edit) in the subsequent New dialog, and whether these appear as nvs (nvi or nvo) or nci. Typically, you repeat this some number of times, until you have defined the necessary collection of nvis, nvos, and ncis that will act as the Lonworks interface.

Note: *If compatibility with Echelon LNS is required, you must create two specific nv entries. See "Local Nvs required by LNS" on page 3-42 for more details.*

Figure 3-53 *New dialog*

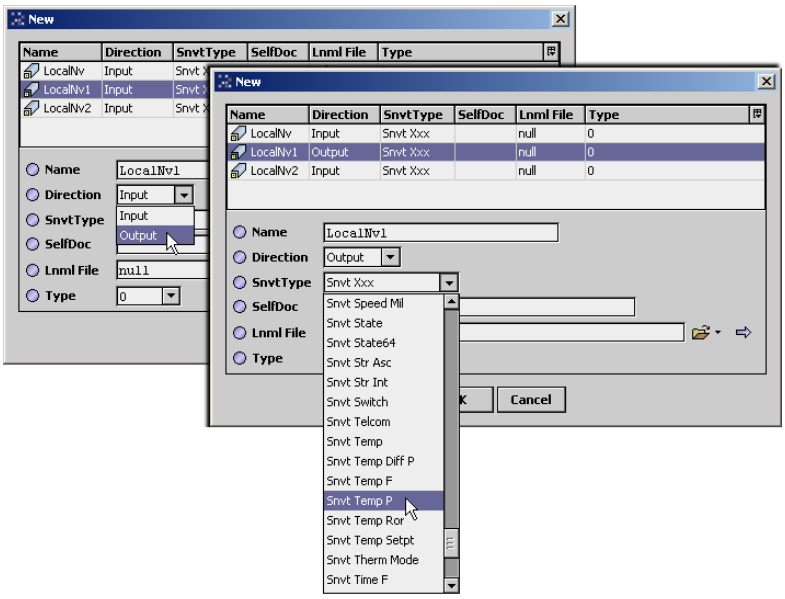


Note: *If you specify* **Local Nci**, *direction is always input, whereas if you specify* **Local Nv**, *you can set each entry as an nvi or nvo, as needed.*

New

In the **New** (or **Edit**) dialog for **Local Nvs**, you specify the *Direction* (input or output) for each entry, the *SNVT type* used, and type in the desired *Name*. The dialog for **Local Ncis** works the same way, except that direction is *fixed* at Input.

Figure 3-54 *Second New dialog, to specify Nvs or Ncis*



Note: *Starting in AX-3.5, as shown in Figure 3-54, the* **New** *(or* **Edit***) dialog includes three additional properties, two of which are "new":*

- **SelfDoc** —Formerly available, but only on the property sheet of the component, once created.
- **Lnml File** — A new property that supports use of manufacturer-defined *user nvs*, or UNVTs (vs. SNVTs) by specifying an lnml file for a device that implements the UNVT needed. In this case, simply leave the **Snvt Type** property at its default "SNVT Xxx" value.
- **Type** — Works in conjunction with the **Lnml File** property, where once you specify an lnml file, the **Type** drop-down control provides a list of UNVTs from which you can choose to use for this nv.

Notes on editing LocalNvs and LocalNcis

- Typically, you change default *names* (`LocalNvn` or `LocalNcin`), editing each one to a unique name using a prefix "type", such as "`nviSetpointA`", "`nvoUnitStatusA`", "`nciOffsetB`", and so on.
- SNVT type selection defaults to "Snvt Xxx" (Type 0, or no SNVT)...typically for any selected item(s), use the drop down list to select a SNVT (list is ordered alphabetically).
  Starting in AX-3.5, instead of a SNVT for an item, you can specify a particular manufacturer-defined user nv, or UNVT, using the **`Lnml File`** and **`Type`** properties. For details, see .
- Entering **`SelfDoc`** (self documentation) for nvs is typically recommended, and in some cases required. A special *syntax* applies. You can omit self documentation by entering a single asterisk (*). For more details, see LNS-compatible nv self documentation syntax.
  Note: *Prior to AX-3.5, the SelfDoc property is not on the New/Edit dialog of an item selected from the Local Nv Manager. Instead, you must access the property sheet of the LocalNv or LocalNci to edit.*

When you click **`OK`**, LocalNvs or LocalNcis are created under the Local Lon Device, and appear listed in the Local Nv Manager.

See the following sections for more details:

- Local Nvs required by LNS
- Local Nv Manager notes

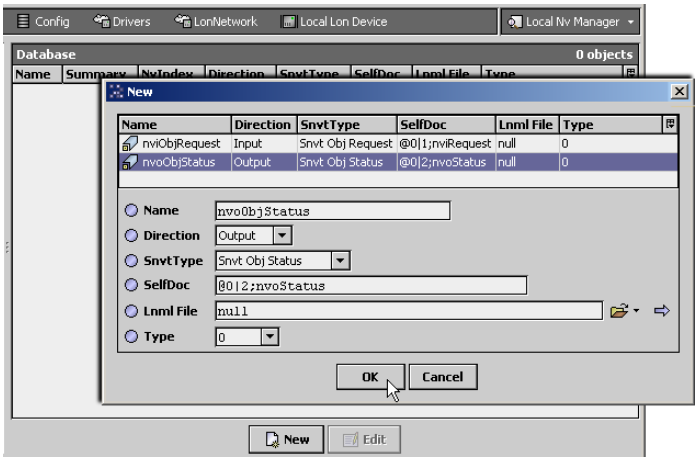
Local Nvs required by LNS

To permit compatibility with an LNS-based network management tool, use the **`Local Nv Manager`** to make two *required* nv entries in the Local Lon Device, as follows:

- `nviObjRequest` (Direction = Input) (Snvt Type = Snvt Obj Request) [SelfDoc = `@0|1;nviRequest`]
- `nvoObjStatus` (Direction = Output) (Snvt Type = Snvt Obj Status) [SelfDoc - `@0|2;nvoStatus`]

Figure 3-55 shows these two specific nvs being added.

Figure 3-55 *LNS compatibility requires first two nvs as shown*



See the next section "LNS-compatible nv self documentation syntax" for further details.

LNS-compatible nv self documentation syntax

All Local Nv or Local Ncis under the LocalLonDevice should have self documentation entered for LNS compatibility, using this syntax:

`@fbIndex|memberNum;text`

where, from left-to-right:

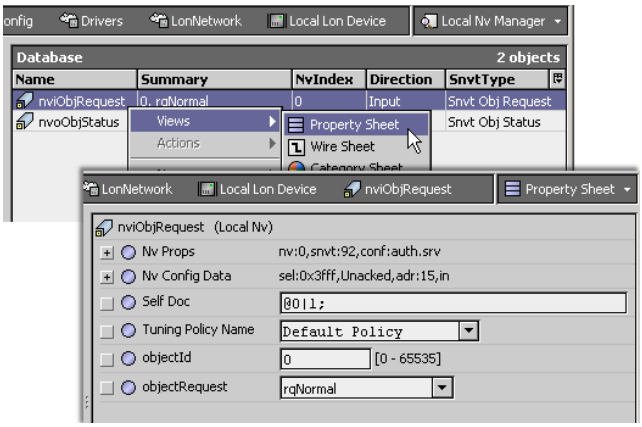
- `@` denotes the start of nv self-documentation.
- *`fbIndex`* is the functional block index.
- `|` is the functional profile separator.
- *`memberNum`* is the member number within the functional block.
- `text` is optional text for describing the nv, by convention "nvi*Something*" or "nvo*Something*"

Note: *Optionally, you can omit self documentation for a Local Nv or Local Nci by simply entering a single asterisk (`*`) instead. However, this does not apply to the two required nvs* `nviObjRequest` *and* `nvoObjStatus`*—for each of these local nvs, you must enter "Self Doc" property values using the syntax above.*

Starting in AX-3.5, you can edit the self documentation (property "SelfDoc") for any Local Nv or Local Nci from its **`New`** (or **`Edit`**) dialog, as launched from the **`Local Nv Manager`**, as shown in Figure 3-55.

Note: *Prior to AX-3.5, access the Self Doc property from the property sheet of the Local Nv or Local Nci, as shown in Figure 3-56.*

Figure 3-56 *Access Self Doc property of LocalNv from its property sheet*



Example

below are two common "Self Doc" property values for the two required Node object nvs:

- nviObjRequest, "Self Doc" = `@0|1;nviRequest`
- nvoObjStatus, "Self Doc" = `@0|2;nvoStatus`

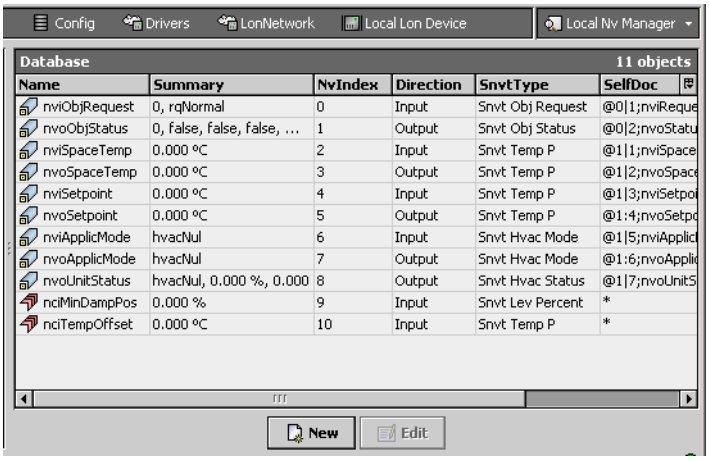
After creating the two required nvs, you can continue defining/adding other local nvs and ncis as needed. Optionally, you can also add "Self Doc" strings to other local nvs and ncis, and also edit the self doc string for the LocalLonDevice (see "LocalLonDevice External Config" on page 3-11).

If you choose to omit self-documentation (for any items except the two Node object 0 nvs described above), simply enter a single asterisk (*) in the "Self Doc" property. Refer to the appropriate LonWorks/ LonTalk reference on the syntax required for nv self documentation.

Local Nv Manager notes

While defining local nvs and ncis, the Local Nv Manager lists items with a sort order of "NvIndex," as shown in Figure 3-57. However, you can click on any column header needed to resort entries.

Figure 3-57 *Default sort of local nvs, ncis, is by NvIndex*



Please note these additional things:

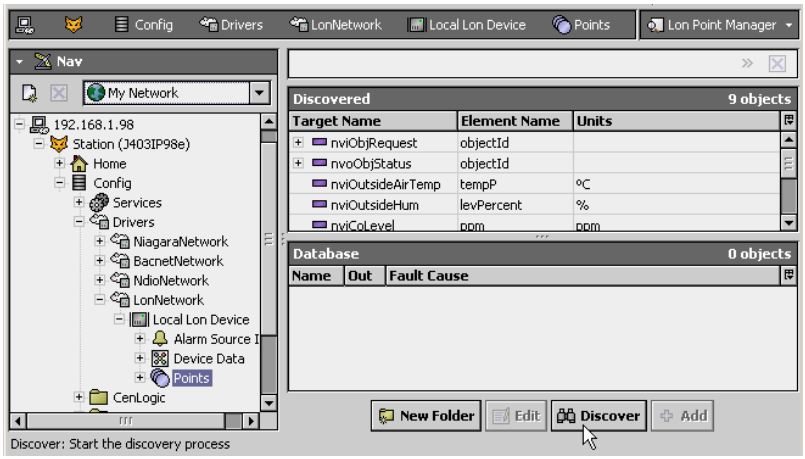
- In general, you should try to *finalize* the creation of the nvis, nvos, and ncis you wish to expose for the station *before* you add associated proxy points (see "Add Local Lon Device proxy points"). This means review and (if necessary) edit the name, direction, and SNVT type for each entry. Note that the **`Edit`** dialog operates like the New dialog for any item.
- Following this initial configuration, if you change (or add to, delete from) the definition of any of these items, you typically need to perform (external) network management on the JACE node to maintain proper network configuration. This includes maintenance of any bindings to the JACE.

Add Local Lon Device proxy points

After you Add Local nvs and ncis using the Local Nv Manager, you can add proxy points under the Local-LonDevice, where each proxy point associates with a specific nv or nci. This allows exchange of station data to those nvs and ncis.

Use the **Lon Point Manager** view (Figure 3-57), which you can access by simply double-clicking the LocalLonDevice's **Points** extension in the Nav side bar.

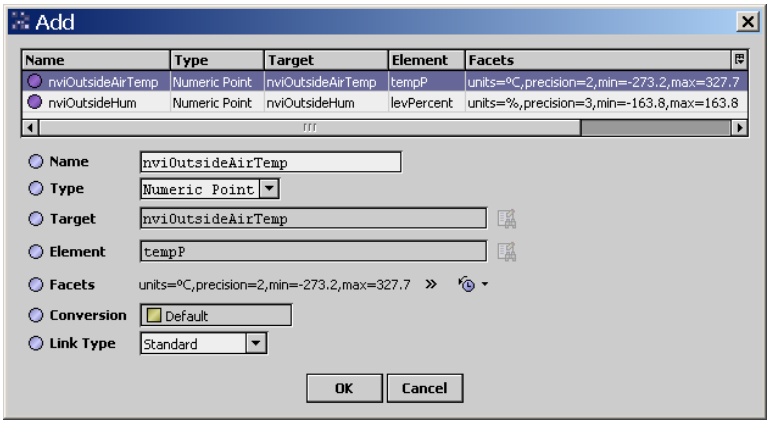
Figure 3-58 *Lon Point Manager of LocalLonDevice*



When you click **Discover** (or toggle Learn Mode on) you can see the collection of nvs and ncis in the discovered data (top pane), listed by name. Any using a structured SNVT will appear with a "+" to the left of the name, showing only the first data element in the structure. Click the "+" to expand for all items in the SNVT structure. Each item (row) is an available candidate for a proxy point.

Click to select one or more discovered items, then click **Add** for the **Add** dialog (Figure 3-58).

Figure 3-59 *Add dialog for LocalLonDevice proxy points*



Local Lon proxy point add notes

Depending on the item selected, the *default* point type will be either writable or read-only, as follows:

- nvi (read-only, that is NumericPoint, EnumPoint, BooleanPoint, or StringPoint)
- nvo (writable, that is NumericWritable, EnumWritable, BooleanWritable, or StringWritable)
- nci (read-only, that is NumericPoint, EnumPoint, BooleanPoint, or StringPoint)

Note: *Point type selections for local nvis and ncis are limited to read-only types (meant to be written externally, that is, outside the station). In the case of local nvos, only writable point types have any practical application--as only the station is capable of writing to its own nvos. Note that this is "the reverse" of the default point type selections when creating other Lon proxy points.*

However, you can select another data category. See "Lon proxy point type selection" on page 3-39.

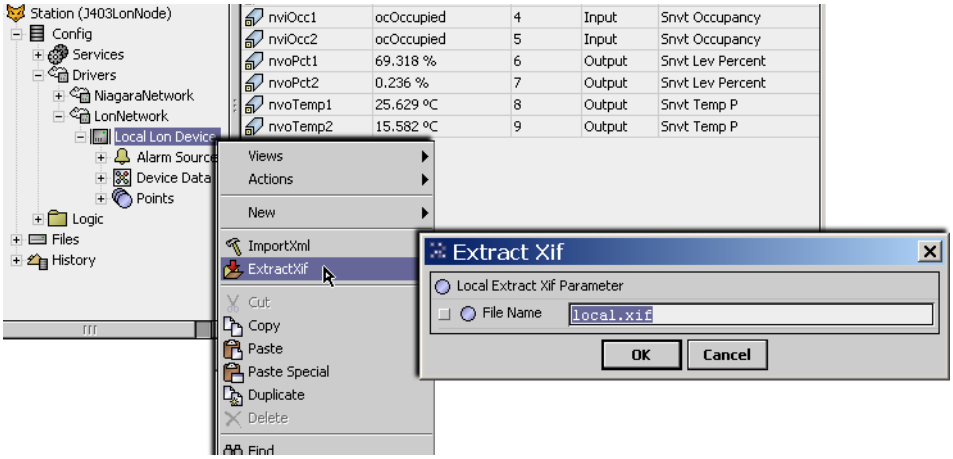
LocalLonDevice XIF generation (AX-3.4 and later)

Starting in AX-3.4, you can use Workbench to create an `.xif` file (external interface file) of a LonNetwork's LocalLonDevice, using a right-click command on the Local Lon Device. The `.xif` file can describe the Lonworks capabilities of the device (JACE) to another LonWorks network management tool, such that offline engineering may be done.

Note: *This should be the "last step" if configuring as a Lon node, after finishing other engineering under the Local-LonDevice (see "Notes when configuring as Lon node" on page 3-40).*

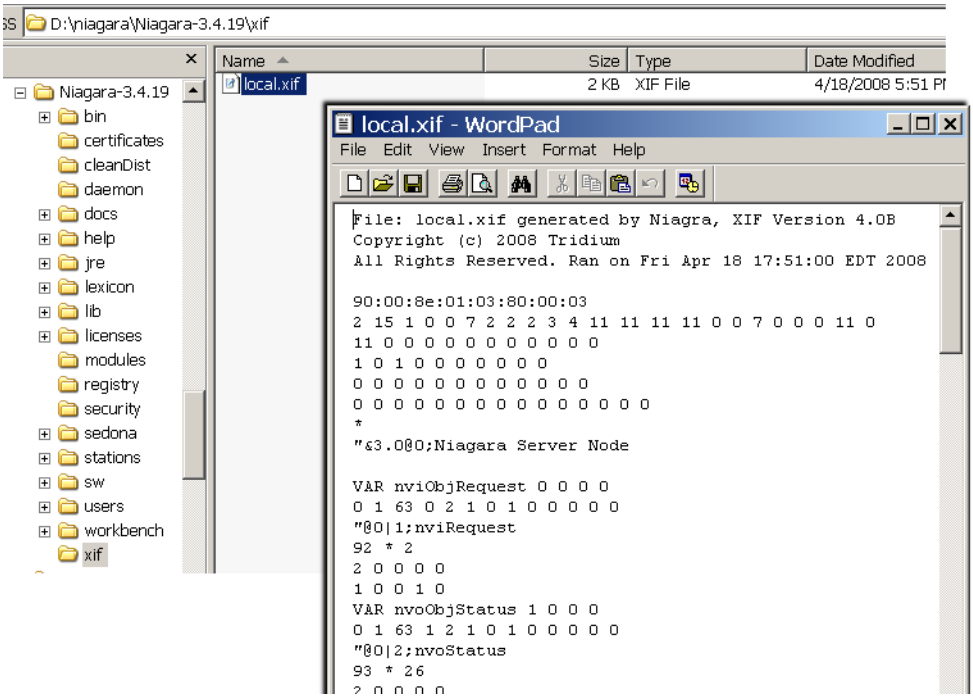
To create the `.xif` file, in Workbench right-click the LocalLonDevice and select **`ExtractXif`** (Figure 3-60).

Figure 3-60 *ExtractXif right-click command on LocalLonDevice*



After typing in the desired name of the .file in the **`Extract Xif`** dialog and clicking **`OK`**, the .xif file is created under a `/xif` folder in the Workbench home directory (`!/xif`).

Figure 3-61 *Example .xif file extracted from LocalLonDevice*



Chapter 4

Lon Plugin Guides

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

Plugins in lonIp module

The lonIp module contains the following plugins (views):

- MemberManager

lonIp-MemberManager

The MemberManager is the default view on the IpChannel component under a LonIpNetwork (starting in AX-3.5 and later). It is used to manage ChannelMember children of the IpChannel's **MemberTable** component.

For more details, see Appendix "Lon over IP", including the section "About the IpLonNetwork's Member Table" on page C-4.

Plugins in lonworks module

The lonworks module contains the following plugins (views):

- ChangeableNvManager
- LinkFilterView
- LocalNvManager
- LonDeviceManager
- LonLinkManager
- LonPointManager
- LonRouterManager
- LonUtilitiesManager
- LonXmlCreate
- NcManager
- NvManager

lonworks-ChangeableNvManager

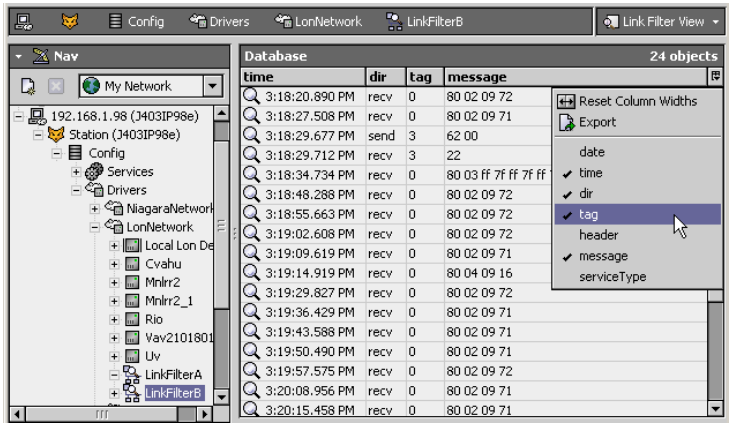
Use the Changeable Nv Manager to review/edit any nvs in a Lon device that are "changeable-types." The Changeable Nv Manager is a view on a LonDevice (providing it supports changeable-type nvs). To view, right-click a LonDevice and select **Views > Changeable Nv Manager**.

For more details, see "About the Changeable Nv Manager" on page 3-32.

lonworks-LinkFilterView

The Link Filter View is the default view on a LinkFilter component, a debug-type object found in the lonworks palette. This table-based view provides low-level messaging data related to Lonworks links.

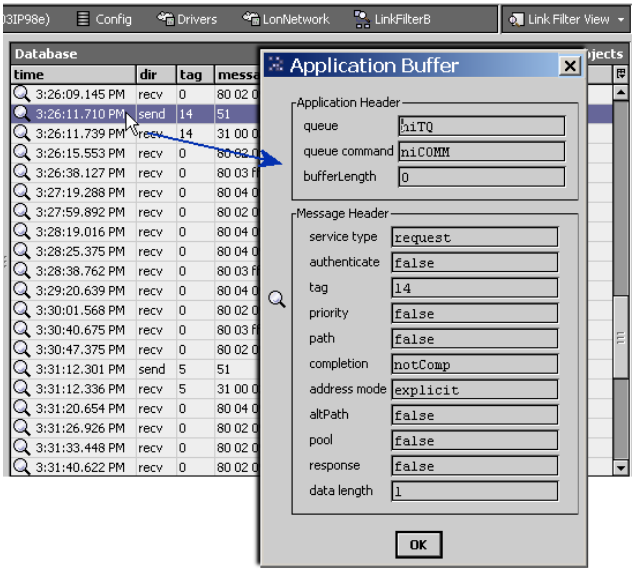
Figure 4-1 *Link Filter View of LinkFilter debug component*



By default, this view shows the time, direction (send or receive), and hexadecimal-encoded link message. As shown in Figure 4-1, you can use the table control to select other columns for display, such as tag, header, and serviceType.

For further debug details, double-click any link event for a popup Application Buffer dialog (Figure 4-2).

Figure 4-2 *Application Buffer details on a selected link event*



See "lonworks-LinkFilter" on page 5-5 for related details on LinkFilter setup.

lonworks-LocalNvManager

Use the Local Nv Manager to add, edit, and access items locally exposed as SNVTs (input or output), applicable only when the station is *not* acting as the Lonworks network manager.

The Local Nv Manager is the default view on the LocalLonDevice. To view, right-click LocalLonDevice and select **`Views > Local Nv Manager`**.

For more details, see "Local Nv Manager" on page 3-12.

lonworks-LonDeviceManager

Use the Lon Device Manager to manage Lonworks nodes (devices) in a Lonworks network, including building new networks (discovering, adding, and commissioning devices) as well as replacing devices. It is the default view on the LonNetwork.

To view, right-click LonNetwork and select **`Views > Lon Device Manager`**.

For more details, see "About the Lon Device Manager" on page 3-14.

lonworks-LonLinkManager

Use the Lon Link Manager to access Lonworks links (bindings) between Lon devices on the Lonworks network. It is one of four views on the LonNetwork.

To view, right-click LonNetwork and select **Views > Lon Link Manager**.

Two tabs are included on this view:

- NetworkVariableLinks
- MessageTagLinks

For more details, see "About the Lon Link Manager" on page 3-20.

lonworks-LonPointManager

Use the LonPointManager to create, edit, access, and delete Lonworks proxy points under a LonDevice. The LonPointManager is the default view for the LonPointDeviceExt (`Points` container) under a LonDevice. The LonPointManager is also the default view for any LonPointFolders under the `Points` container.

To view, right-click LonPointDeviceExt or LonPointFolder and select **Views > Lon Point Manager**

For more details, see "Lon Point Manager tips" on page 3-38.

lonworks-LonRouterManager

Use the Lon Router Manager to manage *Lonworks routers* in a Lonworks network, including building new networks (discovering, adding, and commissioning), as well as replacing routers. It is one of four views on the LonNetwork.

To view, right-click the LonNetwork and select **Views > Lon Router Manager**.

For more details, see "About the Lon Router Manager" on page 3-18.

lonworks-LonUtilitiesManager

The Lon Utilities Manager is one of four views on the LonNetwork. Use the Lon Utilities Manager to access Lonworks diagnostic utilities to apply to selected Lonworks nodes, such as those included in the former NODEUTIL (command line) program. This collection of utilities allow you to query and manipulate the status of a Lonworks node, find nodes on the network, identify nodes, display a node's file structures and data structures, discover more about communications errors, verify binding information, and more.

To view, right-click LonNetwork and select **Views > Lon Utilities Manager**.

For more details, see "About the Lon Utilities Manager" on page 3-25.

lonworks-LonXmlCreate

LonXmlCreate is the plugin name for the Lon Xml Tool, a Workbench utility to make a Lon Xml (.lnml) file for a Lon device from its .xif file. For more details, see "Workbench Tools" on page A-1.

lonworks-NcManager

Use the Nc Manager to browse values of ncis in a Lon device. The Nc Manager is a view on any LonDevice. To view, right-click a LonDevice and select **Views > Nc Manager**.

For more details, see "About the Nc Manager" on page 3-32.

lonworks-NvManager

Use the Nv Manager to browse values of nvis and nvos in a Lon device, or to reassign tuning policies. The Nv Manager is the *default* view on any LonDevice. To view, right-click a LonDevice and select **Views > Nv Manager**, or simply double-click the LonDevice.

For more details, see "About the Nv Manager" on page 3-31.

Chapter 5

Lon Component Guides

Summary information on Lonworks-related components is provided in alphabetical order, for components found in the following modules:

- kitLon
- lonIp
- lontunnel
- lonworks

Components in kitLon module

The `kitLon` module became available starting with AX-3.1, and contains the following components:

- BufferParams (AX-3.5 and later)
- LonPoint
- LonReplace
- LonTime
- LonTodEvent

kitLon-BufferParams

(AX-3.5 and later) BufferParams provides a mechanism to adjust the message buffers in the Neuron chip of the LON adapter of a JACE controller, and works only as a child of the **`LocalLonDevice`** (the only valid parent). Properties allow changing the Application and Network buffer counts and sizes.

Caution *Usage is only for advanced users who understand the implications of changing these buffer settings. Improper usage can degrade operation and/or result in loss of data!*

When copied in the LocalLonDevice, the **`BufferParams`** component dynamically updates its properties to reflect the current message buffer configuration of the Neuron chip. Before changing any buffer parameters, it is recommended to record these default settings.

- To verify the Neuron's current buffer configuration, use the **`Lon Utilities Manager`**, selecting the **`LocalLonDevice`** with Command: `Data Structs`, Subcommand: `Read Only Structure`.
- Upon a property sheet **`Save`**, changes to buffer count or size parameters are reflected in the component's **`Current Size`** property. To be applicable, modifications *cannot exceed* the size shown in the **`Original Size`** property—that is, *Current Size must be less than (or equal to) Original Size*. Therefore, if increasing a buffer size you often need to *decrease* its count, for example.
- Buffer changes saved in the BufferParam's property sheet are *not* automatically written to the Neuron chip in the LocalLonDevice.
  - To write buffer changes to the Neuron, you must issue the action **`Update Buffers`**.
  - Before invoking this action, ensure that Current Size <= Original Size. If Current Size is greater than Original Size, an exception is thrown, and no changes are made to the Neuron chip.

For related details, see "About the Lon Utilities Manager" on page 3-25.

kitLon-LonPoint

(AX-3.3 and later) LonPoint allows you to create a control object that has an nvo with a selectable SNVT type, which can be used to link to an nvi on one or more Lon devices. To use, copy LonPoint from the kitLon palette into the LonNetwork, and select the SNVT type needed. Link the out from a control point (elsewhere in the station) to the dynamic slot of the LonPoint, then link the Nvo slot of the LonPoint to one or more Lon devices.

kitLon-LonReplace

(AX-3.4 and later) LonReplace provides an optional action for a Lon device. To use, copy it from the kitLon palette and paste under a Lon device (or devices), as needed. It provides right-click user access to a **Replace** dialog, allowing a system user to replace the parent device without using Workbench and the Lon Device Manager view. Replacement can use the service pin method or direct entry of the new Neuron ID. Usage may be useful in "appliance scenarios," where system access is done primarily using browser access of Px pages. For related details, see "About Replace" on page 3-16.

kitLon-LonTime

LonTime provides "system time" ability from the station to any Lon device that has an nvi using `SnvtTimeStamp`. To use, copy LonTime from the kitLon palette into the LonNetwork, and link as needed to Lon devices.

kitLon-LonTodEvent

LonTodEvent provides "go-between" ability from BooleanSchedules in the station (outputs: Out, Next Value) to any Lon device that has an nvi using `SnvtTodEvent`. To use, copy LonTodEvent from the kitLon palette into the LonNetwork, and link as needed between BooleanSchedules and Lon devices.

Components in lonIp module

The `lonIp` module became available starting with AX-3.2, and contains a palette with a *modified* LonNetwork (default name is "LonIpNetwork"). The modified LonNetwork communicates on an IP Lonworks channel as defined in CEA-852. The station presents itself as an IP channel node.

Note: *See Appendix "Lon over IP", including "Overview" on page C-1 and "LonIp Setup" on page C-3 for more details. This section simply provides summary descriptions of the different components in that module.*

In the **lonIp** module, the modified LonIpNetwork can contain the following unique components, listed here in alphabetical order:

- ChannelMember
- IpChannel
- IpLocalDevice
- IpLonNetworkConfig
- MemberTable

lonIp-ChannelMember

ChannelMember is a child of the MemberTable container under a LonIpNetwork's IpChannel component. A ChannelMember typically represents a Lon/IP router or an IP channel node.

For details, see "About ChannelMembers" on page C-5.

lonIp-IpChannel

The IpChannel is a child of the LonIpNetwork (copied from the lonIp palette). The IpChannel contains two frozen container slots: IpLonNetworkConfig (**Network Config**) and MemberTable. Starting in AX-3.5, a special (default) view was added for the IpChannel: the **Member Manager** view.

For details, see "About the IpLonNetwork's Member Table" on page C-4.

lonIp-IpLocalDevice

The IpLocalDevice is similar to the LocalLonDevice in a regular LonNetwork, and has that same default name. It is a child of the LonIpNetwork (as copied from the lonIp palette). Use it to specify the station's channel and subnet/node address (it must be consistent with the IP Lon channel that the station is to become a member of). The station presents itself as an IP channel node using this address.

For details, see "LonIp Setup" on page C-3.

lonIp-IpLonNetworkConfig

IpLonNetworkConfig (**Network Config**) is one of two frozen containers under a IpChannel component of a LonIpNetwork. Network Config holds properties that specify the CEA-852 IP Configuration Server (Config Server) and the software ports used.

Note: *Starting in AX-3.5, a property "Is Config Server" is available, with a default value of* `false`*. You can set this to* `true` *to have the JACE station act as the Config Server.*

For details, see "LonIp Setup" on page C-3.

lonIp-MemberTable

The MemberTable is a child of the IpChannel component under the LonIpNetwork. Depending on how the LonIpNetwork is configured, the table is populated in one of two ways:

- AX-3.4 or earlier, or AX-3.5 or later with the IpChannel property "Is Config Server" set to false: After the NiagaraAX device is added in the Config Server, and contact is made with the Config Server (station becomes a member of the CEA-852 channel), this table is dynamically populated with other node members of the channel.
- AX-3.5 or later with the IpChannel property "Is Config Server" set to true:
  In this case you use the (default) "Member Manager" view on the parent IpChannel component to add a new ChannelMember, specifying the IP address of the JACE. You also add a ChannelMember for each Lon/IP router device.

For details, see "About the IpLonNetwork's Member Table" on page C-4.

Components in lontunnel module

The `lontunnel` palette contains the following components:

- LonTunnel
- TunnelService

lontunnel-LonTunnel

LonTunnel is the "server side" component required to support application tunneling of Windows LON-based applications to LON devices reachable in a station's LonNetwork. It is found in the lontunnel module. See "Lon tunneling" on page B-1 for complete information.

For LonTunnel component details, see "LonTunnel component slots" on page B-6.

tunnel-TunnelService

Station server for "application tunneling," where remote PCs with a NiagaraAX Tunnel Client installed can use a legacy or vendor-specific PC application to access devices connected to one or more driver networks. A tunnel connection allows the remote client application to operate as it were directly attached to the driver network (via a "virtual" PC port).

A client PC tunnels using an IP (LAN/WAN) connection, which is granted only after authentication as a station user (with admin write permissions for the particular child tunnel component accessed).

The LonTunnel child component provides support for tunneling Windows LON-based applications. Other serial-based drivers may support a SerialTunnel child component. In any station, only one TunnelService is recommended. It can hold the required number (and types) of child tunnels, as needed.

See "Lon tunneling" on page B-1 for complete information.

Components in lonworks module

The `lonworks` module contains the following components:

- AliasTable
- ConfigParameter
- DeviceData
- DynamicDevice
- LinkFilter
- LocalLonDevice
- LocalNci
- LocalNv
- LonAppDownloadJob
- LonBindJob
- LonBooleanProxyExt
- LonChangeNvTypeJob
- LonCommissioJob
- LonCommissionRouterJob
- LonData
- LonDevice
- LonDeviceFolder
- LonDiscoverJob
- LonEnumProxyExt
- LonFilePos
- LonFileReq
- LonFileStatus
- LonFloatProxyExt
- LonLearnJob
- LonLearnLinksJob
- LonNetmgmt
- LonNetwork
- LonObject
- LonObjectFolder
- LonPointDeviceExt
- LonPointFolder
- LonPollService
- LonReplaceJob
- LonRouter
- LonSetServiceTypeJob
- LonStringProxyExt
- LonTimeZone
- LonTuningPolicy
- LonTuningPolicyMap
- LonWbService
- MessageTag
- NetworkConfig
- NetworkVariable
- RouterEntryTable
- TagLinkEntryTable

lonworks-AliasTable

Contains the data in a Lonworks device's alias table. Appears as a child of the device's DeviceData or ExtDeviceData slot.

lonworks-ConfigParameter

Represents a single config parameter (CP) in a LonDevice. It provides specific support for runtime updates. A node's CPs (if any) appear in the property sheet of the LonDevice.

lonworks-DeviceData

DeviceData is a frozen container slot under most LonDevice objects. It contains the data needed to represent a specific neuron chip including state information and configurable data in the neuron tables as described in "Appendix A: Neuron Chip Data Structures" in the *Neuron Chip Data Book*.

In the case of a LonRouter, two DeviceData container slots exist: "**Near Device Data**" and "**Far Device Data**", representing *both* of the router's neuron chips (relative to the LocalLonDevice).

For more details, see "DeviceData notes" on page 3-30.

lonworks-DynamicDevice

DynamicDevice adds support to LonDevice for dynamically create components from the devices self documentation or a Lon Xml (lnml) file. Almost all Lon devices are implemented as DynamicDevice components.

For more details, see "About Lon devices" on page 3-29.

lonworks-ExtAddressTable

(AX-3.2 and later) Applies only to a device that uses "Extended Network Management" as defined in CEA-709.1-B, and contains the data in its extended address table. Appears as a child of the Lon device's ExtDeviceData slot.

lonworks-ExtDeviceData

(AX-3.2and later) ExtDeviceData is a frozen container slot under any LonDevice that uses "Extended Network Management," as defined in CEA-709.1-B (replacing the more typical DeviceData slot). Like DeviceData, it represents the device's state and configuration data, including the larger number of possible address tables in its child ExtAddressTable slot (unique to ExtDeviceData).

For general information, see "DeviceData notes" on page 3-30.

lonworks-LinkFilter

LinkFilter is a "troubleshooting" component that you can copy from the lonworks palette into a LonNetwork. Its default LinkFilterView provides debug-level messages related to Lonworks links.

To use, you must set *one* of the three "enable" properties to true, and also specify in the corresponding property the target filter, as follows:

- **Enable Subnet Node**
  If true, the filter applies to all but the device specified in that property, using `s/n` numerical syntax. For example:
  Subnet Node = `1/5`
- **Enable Device**
  If true, the filter applies to all but the device named in Device Name property. For example:
  Device Name = `Mnlrr2`
- **Enable Selector**
  If true, the filter applies to all but the device(s) specified by the Selector property, using decimal notation for a selector number (note the Lon Link Manager uses decimal notation for selector numbers, but a Lon device's Nv Manager lists selectors for nvs using hexadecimal numbers). For example:
  Selector = `16`

Collected debug records are then visible in the **Link Filter View** of the component, up the number configured in the **Max Entries** property (default is 1000), after which collection stops. Other boolean properties allow printing results **To Standard Out** of the station, or to **Include Completion Events** (those link events to which Niagara requires notification upon completition).

A "Clear Table" action is provided on a LinkFilter to clear collected records. This re-enables collection if the buffer had previously reached the "Max Entries" limit. If needed, you can copy multiple LinkFilter components in the network in order to configure differently. See "lonworks-LinkFilterView" on page 4-1 for related details.

lonworks-LocalLonDevice

LocalLonDevice represents the local interface to the Lonworks fieldbus. It always appears listed in the LonNetwork.

For more details, see "About the Local Lon Device" on page 3-11.

lonworks-LocalNci

A single network variable in a LocalLonDevice. It is represented in the external interface of the local device as presented to other devices on the lonworks trunk.

lonworks-LocalNv

A single network variable in a LocalLonDevice. It is represented in the external interface of the local device as presented to other devices on the lonworks trunk.

lonworks-LonAppDownloadJob

LonAppDownloadJob is used to command loading a new application image to a dev. table.

lonworks-LonBindJob

Used to command the binding of links between lonworks devices.

lonworks-LonBooleanProxyExt

LonBooleanProxyExt is the Lon proxy extension for a BooleanPoint. It will link a single boolean point to a lonworks LonPrimitive. The appropriate conversions will be performed.

lonworks-LonChangeNvTypeJob

LonChangeNvTypeJob provides feedback on a change nv operation.

lonworks-LonCommissionJob

Used to command that a specific node be commissioned. This will configure the domain table and initialize the address table and nv config table.

lonworks-LonCommissionRouterJob

Used to command the addNode service be performed on the specified node. This service will configure the domain table and initialize the address table and nv config table.

lonworks-LonData

LonData is the superclass for all classes which can be used as the data component of LonComponents. These classes can convert their data from and to the byte format used by physical lonworks device. It will contain one or more LonPrimitives.

lonworks-LonDevice

LonDevice is the base class to represent a physical lonworks device. Adds support for polling, lonLinks and file access for support of configuration properties. Typically, most Lon devices are represented by a DynamicDevice, a component that is subclassed from the LonDevice.

For more details, see "About Lon devices" on page 3-29.

lonworks-LonDeviceFolder

LonDeviceFolder is the Lon implementation of a folder under a LonNetwork. You can use these folders to organize LonDevices in a LonNetwork.

Typically, you add such folders using the **`New Folder`** button in the LonDeviceManager view of the LonNetwork. Each LonDeviceFolder has its own LonDeviceManager view. The LonDeviceFolder is also available in the lonworks palette.

lonworks-LonDiscoverJob

LonDiscoverJob queries the connected network for undiscovered devices.

lonworks-LonEnumProxyExt

LonEnumProxyExt defines a Lon proxy point with a StatusEnum output. The appropriate conversions will be performed.

lonworks-LonFilePos

Represents SNVT_file_pos.

lonworks-LonFileReq

Represents SNVT_file_req.

lonworks-LonFileStatus

Represents SNVT_file_status.

lonworks-LonFloatProxyExt

LonFloatProxyExt defines a Lon proxy point with a StatusFloat output. The appropriate conversions will be performed.

lonworks-LonLearnJob

LonLearnJob is used to command the learnNode service be performed on the specified node. This service will read the domain table, address table and nv config table. No selections - Assumes user is learning a previously managed network. 1) If duplicate subnet/node found abort process. 2) Match devices to database devices with zero neuronId 3) Add dynamic device - if available lnml file import to device otherwise learn nvs from device. Unmanaged devices - same as no selections with no check for duplicate nodes.

lonworks-LonLearnLinksJob

LonLearnLinksJob is used to command the learnNode service be performed on the specified node. This service will read the domain table, address table and nv config table.

lonworks-LonNetmgmt

LonNetmgmt is the base component for all lonworks network management functions. The LonNetmgmt is a frozen container slot of a LonNetwork.

For more details, see "About Lon Netmgmt" on page 3-9.

lonworks-LonNetwork

LonNetwork is the base container for all Lonworks (Lon) components in the station.

Note: *If the host JACE has multiple LON ports, a separate LonNetwork is needed for each physical port. In each LonNetwork, its* **`Lon Comm Config`** *specifies the port's Device Name.*

In addition to being the network container for LonDevices and their child data objects (Lon proxy points), the LonNetwork also contains a LocalLonDevice, which configures the station's representation as a Lonworks device. The LonNetwork also contains a LonNetmgmt component, the container for LON network management functions provided by the station.

As with other NiagaraAX driver networks, the LonNetwork should reside under the station's Drivers container. For general information, see "About Lon Network Architecture" on page 3-6.

The default view of the LonNetwork is the Lon Device Manager.

lonworks-LonObject

LonObject is a container object to allow grouping of LonComponents in a LonMark compliant device. Each LonObject represents a LonMark object, where its child LonComponents represent the set of nvs, ncis and cps in that LonMark object. Starting in AX-3.3, support was added for this feature on a network-wide basis ("Use Lon Objects" property in the network's Lon Netmgmt container), which applies to any "Quik Learn" or "Add" device operation. The "Use Lon LonObjects" option is also available selectively as when issuing an ImportXml command or Learn Nv on a Lon device.

For more details, see "LonObjects" on page 3-36.

lonworks-LonObjectFolder

LonObjectFolder is a folder for use with LonObjects, which starting in AX-3.3 became an optional feature when adding (learning) new Lon devices—as a way to logically group its LonComponents. In any NiagaraAX release, the LonObjectFolder is also used under a *programmable* Lon device, such as an XL15C.

In the scenarios above, you can copy these folders from the `lonworks` palette and place under the device, to further organize LonObjects. Prior to AX-3.3, these folders have no practical application for most devices.

lonworks-LonPointDeviceExt

LonPointDeviceExt is the Lonworks implementation of PointDeviceExt. Its primary view is the LonPointManager. It is the only device extension for any LonDevice. In the lonworks palette, it is available under the `DynamicDevice` as `Points`.

lonworks-LonPointFolder

LonPointFolder is the Lonworks implementation of a folder under a LonDevice's Points container (LonPointDeviceExt). You add such folders using the **`New Folder`** button in the LonPointManager view of the Points component. Each LonPointFolder has its own LonPointManager view. The LonPointFolder is also available in the lonworks palette.

lonworks-LonPollService

The LonPollService provides a concrete class for polling lonworks devices. For more details, see "Lon Poll Service notes" on page 3-7.

lonworks-LonReplaceJob

LonReplaceJob is used to command that a specific node be replaced. This will set the domain table and address table and nv config table to match configuration of the original device.

lonworks-LonRouter

LonRouter represent a Lonworks router. Typically, you create and manage LonRouters from the LonRouterManager view of the LonNetwork. The LonRouter is also available in the lonworks palette.

lonworks-LonSetServiceTypeJob

LonSetServiceTypeJob implements the SetServiceType action for LonNetmgmt.

lonworks-LonStringProxyExt

LonStringProxyExt is the Lon proxy extension for a StringPoint. It will link a single string point to a lonworks LonPrimitive. The appropriate conversions will be performed.

lonworks-LonTimeZone

Represents SNVT_Time_Zone.

lonworks-LonTuningPolicy

A tuning policy for the LonNetwork, with standard NiagaraAX tuning policy properties. See "Lon Tuning Policies" on page 3-8.

lonworks-LonTuningPolicyMap

LonTuningPolicyMap (Tuning Policies) is the LonNetwork container slot for one or more LonTuningPolicy(ies).

lonworks-LonWbService

LonWbService provides access to a Lonworks network for client-side applications.

lonworks-MessageTag

MessageTag represents a single network variable in a LonDevice. It provides specific support for runtime updates and contains data needed to support network management. The MessageTag is available in the lonworks module.

lonworks-NetworkConfig

NetworkConfig represents a single NCI in a LonDevice, as one type of LonComponent. It provides specific support for runtime updates and contains data needed to support network management.

For more details, see "About LonComponents" on page 3-35.

lonworks-NetworkVariable

NetworkVariable represents a single network variable in a LonDevice, as one type of LonComponent. It provides specific support for runtime updates and contains data needed to support network management.

For more details, see "About LonComponents" on page 3-35.

lonworks-RouterEntryTable

RouterEntryTable is a table in a Lonworks router.

lonworks-TagLinkEntryTable

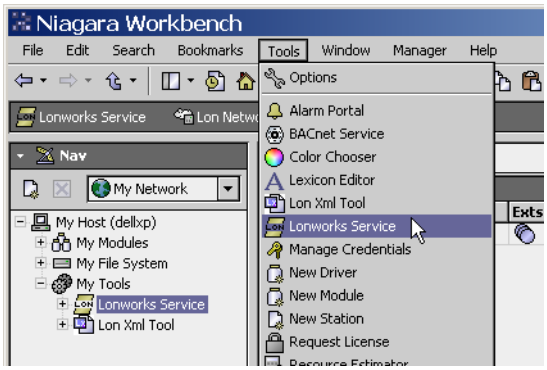
TagLinkEntryTable is a table for Lonworks message tags.

Appendix A

Workbench Tools

Included in Workbench's menu are various "tools," some of which apply directly to Lonworks.

Figure A-1 *Tools menu in Workbench*



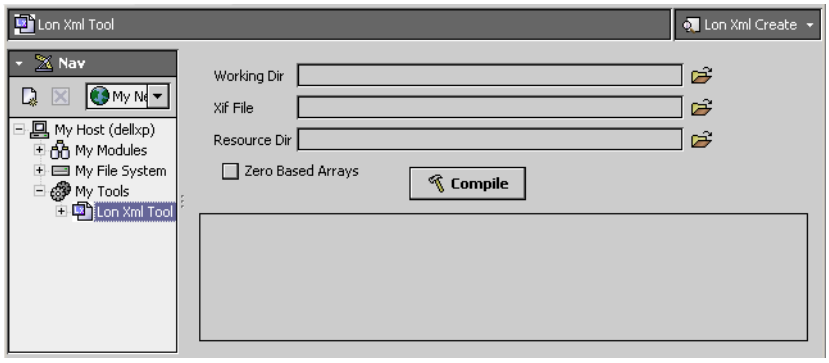
The following Lonworks Workbench tools are included:

- Lon Xml Tool
- Lonworks Service

Lon Xml Tool

The **`Lon Xml Tool`** lets you make your own Lon Xml (.lnml) file for a device, using the source .xif file and (if necessary) other resource files, as available from the device's manufacturer. The tool's dialog (Figure A-2) provides the necessary input fields.

Figure A-2 *Lon Xml Create tool dialog*



The following sections provide more Lon Xml Tool details:

- Need for custom Lon Xml files
- Lon Xml file overview
- Lon Xml creation
- Storing lnml files
- Differential temperatures and lnml file edits

Need for custom Lon Xml files

Although Workbench installs with many lon<*Vendor*> modules, where each one contains numerous lnml files for specific devices, in some cases you may need an lnml file for a device not included. Typically, this need arises because:

- The device includes one or more manufacturer-defined nvs (otherwise, you could simply use a DynamicDevice to represent it, then perform a Learn Nv action).
- The device is newer or otherwise different than the one represented by the closest "standard" lnml file, or Tridium has not yet released a standard lnml version for this device.

Note: *Lon Xml supports "default" (init) values for a device's ncis and cps, providing that the source .xif file is version 4.0 or later, and "init" data values other than all zeros exist.*

In general, using the Lon Xml Tool allows you to independently integrate Lonworks devices without depending on Tridium to issue updated lon<*Vendor*> modules.

Lon Xml file overview

Lon Xml (lnml) files are used to describe specific Lonworks *devices* to NiagaraAX. This includes data to describe the device itself (typically derived from an *xif file*), as well as manufacturer-specific *datatypes* for nvs, ncis, and cps (typically derived from *resource files*). For specific low-level details about Lon Xml, see the "Lon Markup Language" section in the *Developer Guide*.

Note: *By convention, any device's lnml file is named identically as its xif file (except for extension). Meaning, a device that has a "`T7350Cs.xif`" file will have a "`T7350Cs.lnml`" file.*

By default, Workbench installs with many "standard" lnml files for various Lon devices, all of which are distributed among various lon<*Vendor*> module jars (for example, `lonHoneywell`). During a Lon Device Manager "Discover", these lnml files are automatically searched and defaulted, according to the Program Id returned by each device.

Or, if the needed lon<*Vendor*> modules are installed on the host JACE platform, a "Quik Learn" automatically uses the matching lnml file to create Lon devices, in much the same manner. For related details, see "Lon Device Manager key points" on page 3-14, and "About Quik Learn" on page 3-16.

Note that Lon Xml file source data is required to make lnml files.

Lon Xml file source data

Xif and resource files for any Lonworks device are provided *by the manufacturer*. To create an lnml file for any Lonworks device, you *must* have its xif file. If the device contains manufacturer-defined datatypes, and you want this data for any LonComponents, you must *also* supply related *resource files*.

Resource files typically include the following types:

- *vendor*.ENU — Language resource file (in this case, English).
- *vendor*.FMT — Format selector file.
- *vendor*.FPT — Functional profile file.
- *vendor*.TYP —Type and enum file.

Before using this tool, place all related resource files in a single directory.

Lon Xml creation

When using the Lon Xml Tool tool to make an lnml file, you specify the following, by clicking the folder icon beside each input field (see Figure A-2):

- Working Dir — where the device lnml file is to be made (**`Directory Chooser`** appears).
- Xif File — the manufacturer's xif for that device (**`File Chooser`** appears).
- Resource Dir (if applicable) — the directory that contains associated *resource files* (as listed in the previous section, "Lon Xml file source data") (**`Directory Chooser`** appears).

In the **`Directory Chooser`** or **`File Chooser`** dialogs, you can specify the complete path (inclusive of drive letter) for the working directory, *.xif file, and resource files directory. Or, use the "Sys Home" ("!") path if you are working under the `Niagara-3.n.n` installation directory.

Note: *Starting in build 3.3.25 and in AX-3.4, a "Zero Based Arrays" checkbox can be used to force the use of "zero-based" names for the unique name of each element in an arrayed nv, nci, and cp in the lnml file. Names of arrayed elements are created by appending the array index to the array name. Note the default behavior (unchecked) continues to use "one-based" indices for arrayed nvs, ncis, and cps.*

When you click **`Compile`**, the xif file for that device is parsed, along with the resource files (if applicable), and an *.lnml file for the device is created in the working directory. If using resource files, a "datatypes" *subdirectory* is also created under that working directory, with a child lnml file. See "Storing lnml files" on page A-4.

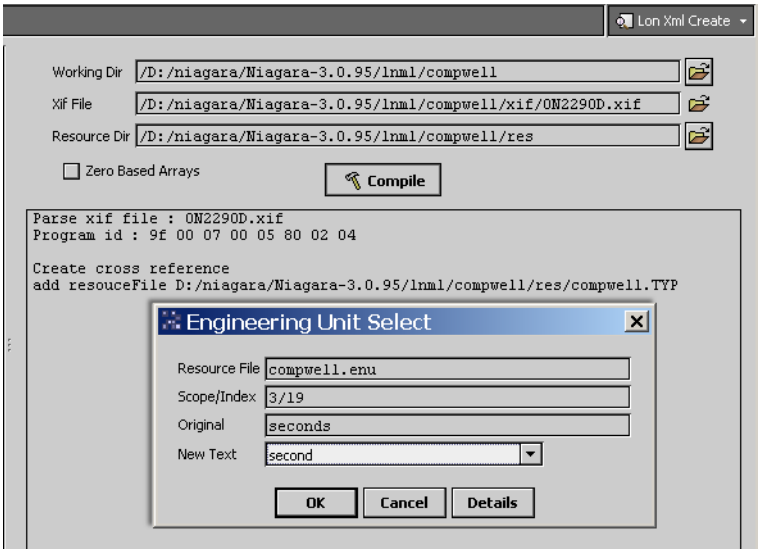
Note: *If using resource files, a popup* **`Engineering Unit Select`** *dialog may appear before the lnml files are created. This dialog lets you review unit strings in the "language resource file" that may need association with Niagara engineering units. See "Engineering Unit Select" on page A-3.*

Engineering Unit Select

Engineering units may be specified for elements in some of the datatypes. Unit descriptor strings are specified in the language resource file (*.enu for English). Strings for units in the resource files are intended for "display only," and do not have a defined format. Whereas, units specified in the lnml file correspond to Niagara units (BUnits), and as such, must map to a specific set of strings.

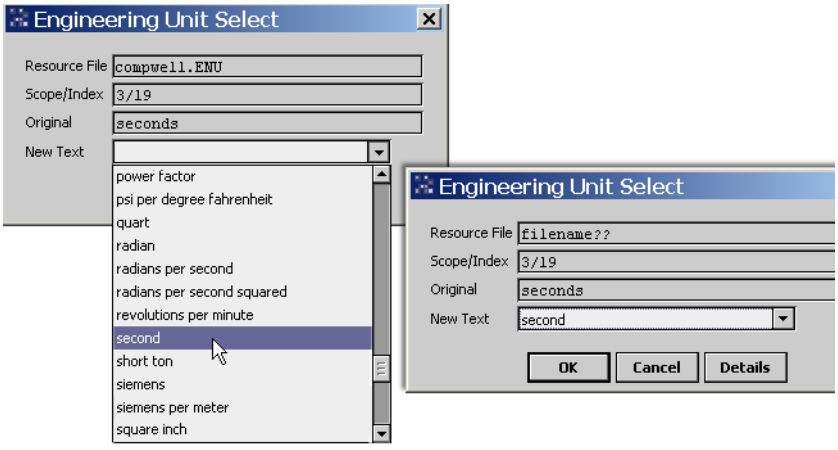
If, when compiling, an exact match is *not* found, a popup dialog shows the original string and lets you pick a new unit from the approved "**`New Text`**" list. A "best guess" entry often exists (Figure A-3).

Figure A-3 *Engineering Unit Select popup dialog when compiling*



In this dialog, the "Scope/Index" field shows the mapping within the language resource file to the unit string, and the "Original" field shows the found string. Typically, the original string is "close" to an approved Niagara unit in the list (possibly the "best guess"), which you select and click **`OK`** (Figure A-4).

Figure A-4 *Selecting unit from drop-down list*



After clicking **`OK`**, if other unit strings are found that need specifying, the dialog remains open—you repeat selections until all unit strings are associated. The device lnml file is then created, along with a datatypes lnml file under the "datatypes" subdirectory of the working directory.

Note: *In a few cases, selection of a Niagara unit based on the "Original" text string may be difficult (does not suggest a normal engineering unit). In this case, you can leave the "New Text" field blank and click* **`OK`** *to go to the next unit. No units will be applied to elements using this string.*

If you are creating lnml files for *multiple* devices by the same vendor, after doing this process for the *first* device, you might not be prompted again with an Engineering Unit Select popup, unless another device (xif) points to a new unit string.

Storing lnml files

Whenever using lnml files made by the Lon Xml Tool, you must preserve the name/subdirectory structure in order to be useful to Workbench. This is particularly important for any lnml file that compiled with a "datatypes" subdirectory. Be sure to move its datatypes subdirectory (and contents) along with any "device" lnml file.

Note that a "device" lnml file is given a name derived from the xif file, and its datatype lnml file(s) are given names derived from the resource files. For example, a lnml file created with source files:

- Xif File: `ON2290D.xif`
- Resource Files:
  - `compwell.ENU`
  - `compwell.FMT`
  - `compwell.FPT`
  - `compwell.TYP`

Will have a file name and subdirectory structure like this:

`ON2290D.lnml`, with subdirectory \ file: `datatypes\compwell.lnml`

Differential temperatures and lnml file edits

In some cases, LonComponents (nvs, ncis, cps) in Lonworks devices have temperature values that are intended as differential ("delta," or "offset") temperatures. This can be an issue when Niagara performs unit conversions, as "absolute" values use a conversion offset, and differential values do not.

Engineering unit support in Niagara distinguishes between absolute temperatures (e.g. "celsius" or "fahrenheit") and *differential* temperatures (e.g. "*degrees celsius*" or "*degrees fahrenheit*").

There are special SNVT types for differential temperatures (e.g. SNVT_temp_diff_p), but some controllers were designed before these types were introduced. As a consequence, these devices may have certain nvs defined as absolute temperatures, when in fact they are intended to be differential.

For unit conversion purposes, if you know which items (nvi, nvo, nci) in a Lonworks device should be treated as a differential temperature, you can *edit* its corresponding device lnml file in a text editor. Using this edited lnml file in Workbench to define Lonworks devices ensures that unit conversions are performed correctly.

Do this for any affected nv or nci by adding "`+diff`" to the `snvtType` attribute, as shown:

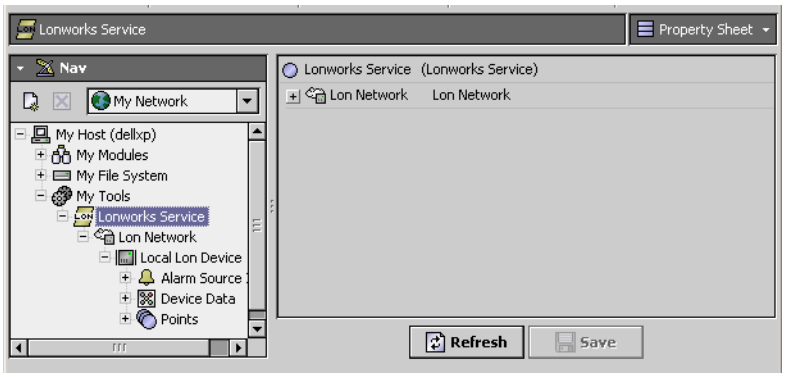
- from: `<snvtType v="TempP" />`
- to: `<snvtType v="TempP+diff" />`

As an alternative to editing device lnml files, you can also change the facets/units in associated Lon proxy points under a device. See "Differential temperature notes" on page 3-4.

Lonworks Service

The Lonworks Service is applicable if your Workbench PC has a Lonworks adapter. It provides you the identical "LonNetwork" access as if you had a local (PC) station running (Figure A-5).

Figure A-5 *Lonworks Service tool in Workbench*



For example, you can use the Lon Device Manager to discover, add, and upload Lon devices, examine nvs and ncis as LonComponents, and even perform writes and do other configuration (make bindings between devices, commission devices, etc.).

Caution *Any configuration you perform is not stored (persisted) in a station database, as this is "station-less" access (modeled completely in RAM). When you close Workbench, all modeling is lost—consider the Lonworks Service like a "hand held device" in this respect.*
For this reason, do not use this tool to access a Lonworks network already managed by a station, but instead open that station, and access its LonNetwork.

Managing Workbench services

The Lonworks Service, like a few other tools, is managed from the **`Workbench Service Manager`**. In that Workbench Tools view, you can start and/or stop the Lonworks Service. You must start the LonWorks Service in order to use it. See the "Workbench service manager" section in the *User Guide* for related details.

Appendix B

Lon tunneling

A NiagaraAX station running the Lonworks driver can provide "tunneling" access to its connected Lonworks devices. This allows you to use a vendor's Windows LON-based application (via the Lon tunnel client) to perform device-specific operations. Examples include application downloads or other device configuration.

The tunneling client is separate from NiagaraAX Workbench—meaning that you can install it on various PCs, as needed. The key advantage is that Lon tunneling requires only a standard IP connection (to the station), yet provides access as if the client PC was attached to the target Lonworks network via a physical FTT-10 adapter.

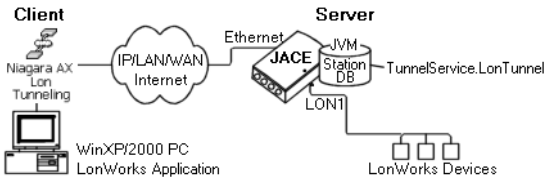
Note: *No special licensing is required to use tunneling features in NiagaraAX.*

The following sections provide more details:

Lon tunnel overview

As shown in Figure B-1, tunneling in NiagaraAX uses a client-server architecture.

Figure B-1 *Lon tunnel architecture*



Lon tunneling uses the following components:

- **Client (PC application) side**
  The **`NiagaraAX Lon Tunnel`** client installs on any Win32-based PC (independent of Niagara Workbench). It provides a "Niagara AX Lon Tunneling" applet in the Windows Control Panel, corresponding to a "virtual" LON port. For details, see "Client side (PC application)" on page B-2.
- **Server (station) side**
  To support Lon tunneling, the host JACE must have the "`lontunnel`" module installed. In addition, its station must be configured with a **`TunnelService`** and a child **`LonTunnel`** component. For details, see "Station side (TunnelService)" on page B-4.

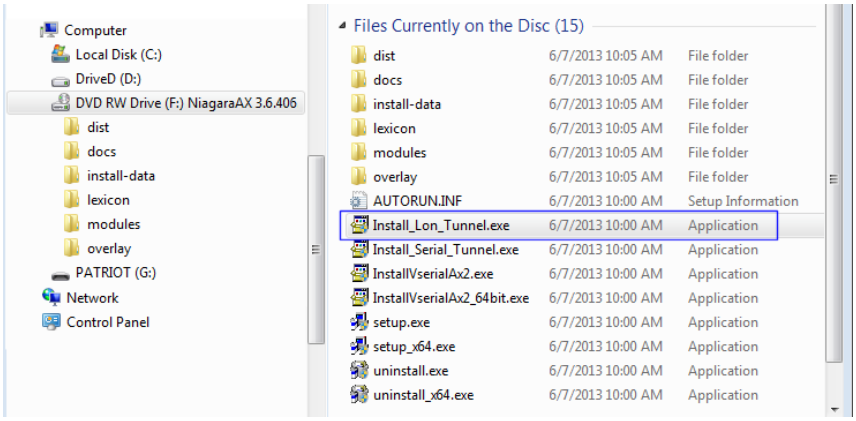
Note: *A SerialTunnel ("`tunnel`" module) is also available, and uses the same basic architecture—as a child under a station's TunnelService. It allows application tunneling to connected devices on most driver networks using RS-232 or RS-485 communications. For details, see "Serial tunneling" in the* Drivers Guide.

Note client tunnel connections (Lon or serial) to a station's TunnelService use basic authentication *(credentials* not *encrypted), so security considerations apply. See "Best security practices for tunneling" on page B-5.

Client side (PC application)

The Lon tunnel client is a self-installing executable: `Install_Lon_Tunnel.exe`, found in the root of the NiagaraAX distribution CD (Figure B-2). As needed, you may install it on any Windows PC on which you use a Windows LON-based application.

Figure B-2 *Lon Tunnel Client installation file is in NiagaraAX CD root*



See the following additional sections:

- Installing the Lon tunnel client
- Lon tunnel client configuration
- Lon tunnel client installation details

Installing the Lon tunnel client

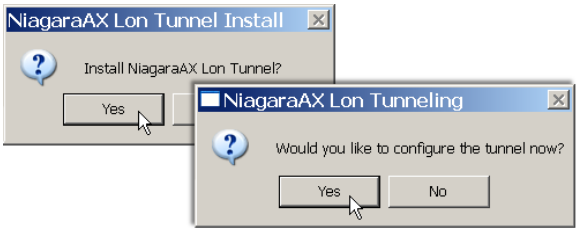
To install the Lon tunnel client

To install the Lon tunnel client, at the Windows XP/2000 PC do the following:

Step 1 Access the file `Install_Lon_Tunnel.exe`, as found on the NiagaraAX CD (Figure B-2).

Step 2 Double-click this file to launch the installation.

Click **`Yes`** to install (a command prompt window opens), and other popup appears.



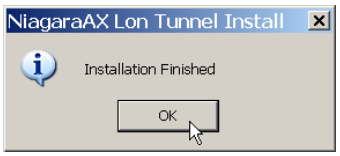
Click **`Yes`** again to configure.

Step 3 In the **`Niagara AX Lon Tunnel`** dialog, enter any known information, or simply accept defaults. If your PC has LON adapter, make sure not to duplicate its `LONn` address. You can always reconfigure using this dialog, by returning via the Windows **`Control Panel`**.

See "Lon tunnel client configuration" on page B-3 for details on all fields in this dialog.

Step 4 Click **`OK`** to finish the install.

The "Installation Finished" dialog appears—click **OK** again.

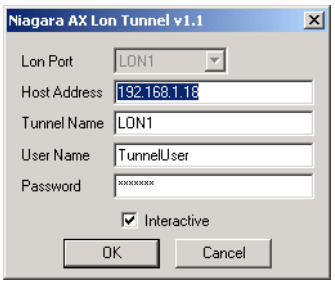


See "Lon tunnel client installation details" on page B-3 for a listing of installed components.

Lon tunnel client configuration

After installation, access the Lon tunnel configuration dialog from the Windows **Control Panel**. As shown for an example session in Figure B-3, all fields require a valid entry.

Figure B-3 *Lon tunnel client session example*



Fields in this dialog are described as follows:

- **Lon Port**
  The "virtual" LON port provided by this tunnel client. This should not conflict with any existing LON port assignment, as known to Windows, for a physical LON adapter (e.g. LON1).
  When you tunnel from a LON-based Windows application, you specify this "virtual" LON port.
- **Host Address**
  The IP address (or hostname) of the tunnel server, meaning the target JACE running a station with a LonNetwork, TunnelService, and LonTunnel.
- **Tunnel Name**
  The `LONn` device name (identifier) of the JACE's LonNetwork to access. This is `LON1` in most cases (any JACE with only a single LON port). However, if the target JACE host has *multiple* LON ports (supports multiple LonNetworks), you could enter `LON2`, for example.
- **User Name**
  User in the target JACE station, where this station user must have *admin write* permissions for the station's TunnelService and child LonTunnel(s).

Caution *The TunnelService uses "basic authentication" for login on any client connection (Lon tunnel or serial tunnel), so we recommend you create a special user in the station to use (only) for all Lon or serial tunnel access. For configuration details, see "Best security practices for tunneling" on page B-5.*

- **Password**
  Password for this station user.
  Note: *The password is* not encrypted *when passed to the station (see Caution above).*
- **Interactive (checkbox)**
  If checked, this dialog reappears each time a LON-based application first opens this "virtual" LON port. If cleared, this dialog displays only if an open fails to establish a connection to the tunnel server (as stored from last entry). Typically, you leave Interactive checked.
  Note: *When this dialog appears interactively, the* **Lon Port** *setting is read-only. To change that, you must access the Lon tunneling applet from the Windows* **Control Panel**.

Lon tunnel client installation details

You can install a newer version of the Lon tunnel client without uninstalling the current version. The following files are installed, with services referenced in the Windows registry:

- Control Panel
  <WINDOWS_SYSTEM_DIR>\vlonax.cpl
- Network Tunnel Service

<WINDOWS_SYSTEM_DIR>\vlonax.exe
Service name: vlonaxSvc (vlonax dependency)

- Lon Driver Service
  <WINDOWS_SYSTEM_DIR>\vlonax.sys
  Service name: vlonax
- Uninstaller
  <WINDOWS_SYSTEM_DIR>\vlonaxun.exe
  Executable via "Add/Remove Programs" in Windows Control Panel

Note: *If uninstalling (**Start > Control Panel > Add or Remove Programs**), and the uninstall appears to fail, try reinstalling the tunnel client, and then uninstall again.*

Station side (TunnelService)

To be a Lon "tunnel server," a JACE must have the lontunnel module installed, and its station must have a TunnelService (under its **Services** folder), as well as a child LonTunnel component.

The following sections provide more details on the "station side" of Lon tunneling:

Configuring the Lon tunnel server

To configure the station for Lon tunneling

To configure the station for Lon tunneling, do the following:

Step 1 In the palette side bar, open the **lontunnel** palette.

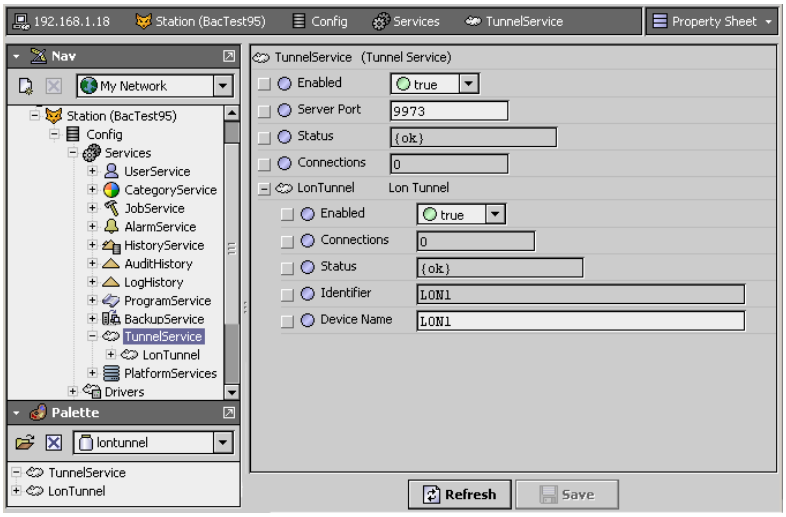
Step 2 Open the JACE station and expand its **Services** folder.

- If no **TunnelService** exists, paste one from the palette into the **Services** folder.
- If a TunnelService does exist, go to next step.

Note: *Only one TunnelService is needed in a station's services, and it can hold multiple tunnel components (both LonTunnel and SerialTunnel). The TunnelService in the **lontunnel** module is identical to the TunnelService in the (serial) **tunnel** module.*

Step 3 From the palette, paste a **LonTunnel** under the station's **TunnelService**.

Figure B-4 *LonTunnel under station's TunnelService*



The station should now have a TunnelService and child LonTunnel component, as shown in Figure B-4 above. See "LonTunnel component slots" on page B-6 for details on various properties.

Note: *If the JACE has multiple LON ports (and corresponding LonNetworks), you can copy the same number of LonTunnel components under the TunnelService. You can then associate each LonTunnel with a particular LonNetwork (by its LONn device name).*

Step 4 **Save** the **TunnelService** configuration when done.

A station user requires *admin write* permissions for the LonTunnel(s) to allow tunnel client access.

Note: *Clients that access the TunnelService (both LonTunnel and SerialTunnel) use "basic authentication" for login access. So in either of these client connections, the user credentials passed to the station are not encrypted—a potential security issue! See "Best security practices for tunneling" on page B-5.*

Also, consider disabling *the TunnelService (set its* Enabled *property to* `false`*) whenever it is not needed.*

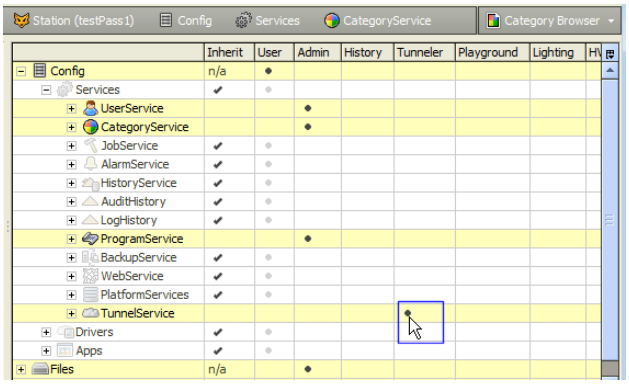
Best security practices for tunneling

Although convenient, Lon or serial tunneling access to a station presents a potential security issue, as a station's **`TunnelService`** uses "basic authentication" for client access to the station. This differs from normal user access (via FoxService and/or WebService), typically using much stronger authentication.

As a workaround, we strongly recommend that you assign the station's TunnelService to a special category *not assigned to any other component* in the station, and create a *special user* that has admin write permissions on *only that single category* (unlike any other user). That should be the *only user* used to make tunnel client connections. See "To configure for safer tunneling access".

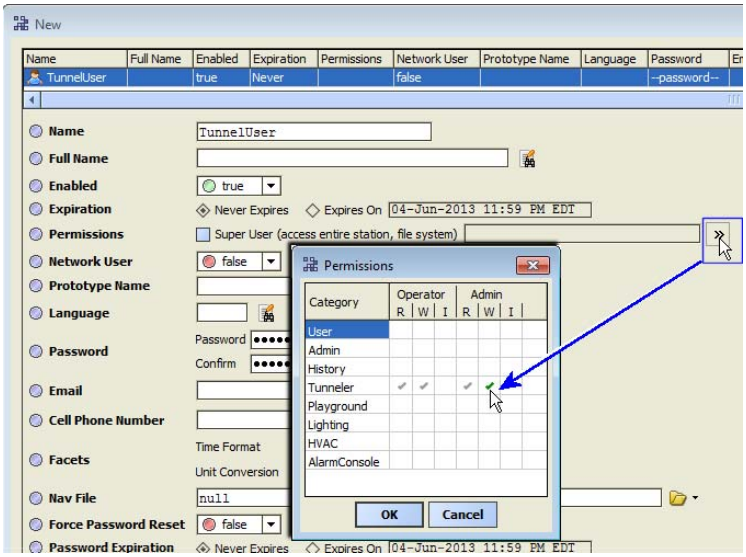
To configure for safer tunneling access

Step 1 In the station's **`CategoryService`**, set up a Category *unassigned to any other component.*



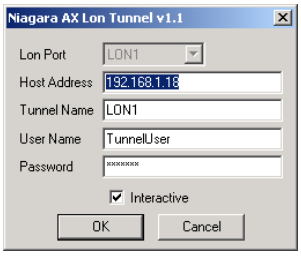
Assign the station's **`TunnelService`** to that category, as shown above.

Step 2 In the station's **`UserService`**, create a new user that has permissions *only* on that one category.



Assign this new user *admin write* permissions to that *one* category, and **`Save`** that user.

Step 3 From any client to the TunnelService (Lon tunnel *or* serial tunnel), only use this special user account.



This workaround provides full tunneling capability, but minimizes the security risk in case the credentials for this special user become compromised.

LonTunnel component slots

Unless you have *multiple* LonTunnels (one for each LON port on the JACE), you typically do not need to do any further configuration, apart from defaults (as shown in Figure B-4 on page B-4).

Properties of the LonTunnel are described as follows:

- **Enabled**
  Boolean slot that must be enabled (true) to permit tunneling.
- **Connections**
  Read-only slot to indicate the number of tunnel connections, as either 0 (none) or 1 (maximum).
- **Status**
  Read-only status of the Lon tunnel, typically `ok` (unless `fault` for no supporting LON port).
- **Identifier**
  Read-only "reflection" of the entered Device Name (below), used as the "Tunnel Name" when configuring the client-side Lon Tunneling dialog (Figure B-3).
- **Device Name**
  String to identify the JACE's `LONn` port used in a LonNetwork. For a LonTunnel copied from the palette, this defaults to `LON1`. These must match the "Device Name" string in the LonNetwork's Lon Comm Config container. See "Lon Comm Config notes" on page 3-6.

In addition, a LonTunnel has an available (right-click) *action*:

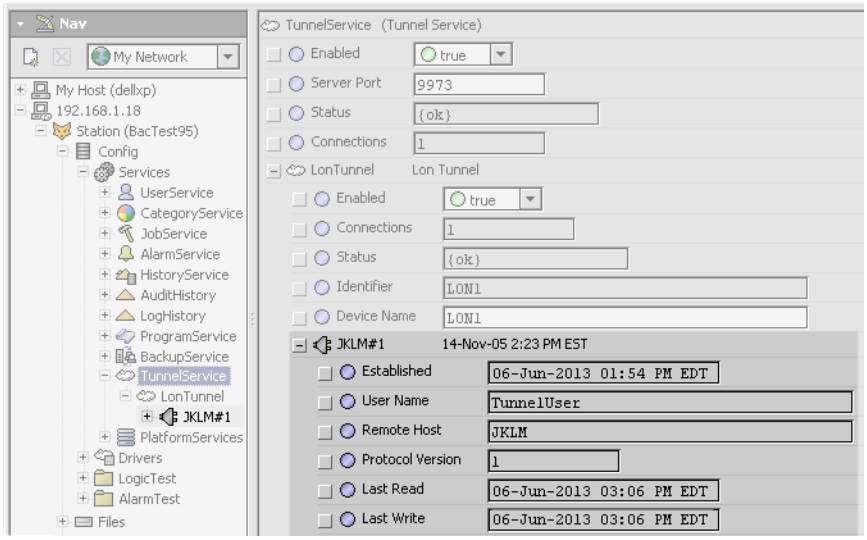
- **Disconnect All**
  Disconnects any active connection through this LonTunnel (maximum of 1), causing removal of the "TunnelConnection" below it. See "About Lon tunnel connections" on page B-6. On the remote (Lon tunnel client) side, a popup message "Connection closed by remote host" is seen.
  Note: *Any TunnelConnection component also has its own "Disconnect" action, which effectively performs the same function.*

About Lon tunnel connections

Under any LonTunnel, only one tunnel connection is supported at any one time—if a tunnel connection is active and another tunnel client (PC) attempts to connect, that remote user sees a popup dialog saying that the "Desired tunnel is busy."

In the station (tunnel server), any active tunnel connection results in a "Tunnel Connection" child component, named as the remote (client's) hostname or IP address, with a "`#1`" suffix (Figure B-5).

Figure B-5 *TunnelConnection is dynamically created/removed*



In the example above, the remote host that is currently Lon tunneling is "JKLM." When a tunnel connection is terminated, this Tunnel Connection component is removed.

Properties of a TunnelConnection (either LonTunnel *or* SerialTunnel) are all read-only, as follows:

- **Established**
  Date-timestamp of when this tunnel connection was first established.
- **User Name**
  User in the station that is currently tunneling.
- **Remote Host**
  Windows hostname of the remote tunneling client.
- **Protocol Version**
  Version of the (remote) Lon tunneling client application being used.
- **Last Read**
  Date-timestamp of when the last read of a station item occurred over the tunnel connection.
- **Last Write**
  Date-timestamp of when the last write to a station item occurred over the tunnel connection.

In addition, a TunnelConnection has an available (right-click) *action*:

- **Disconnect**
  Disconnects the active tunnel connection, removing the parent TunnelConnection component. This causes a popup "Connection closed by remote host" to be seen on the client tunnel side.
  Note: *A LonTunnel component also has its own "Disconnect All" action, which effectively performs the same function.*

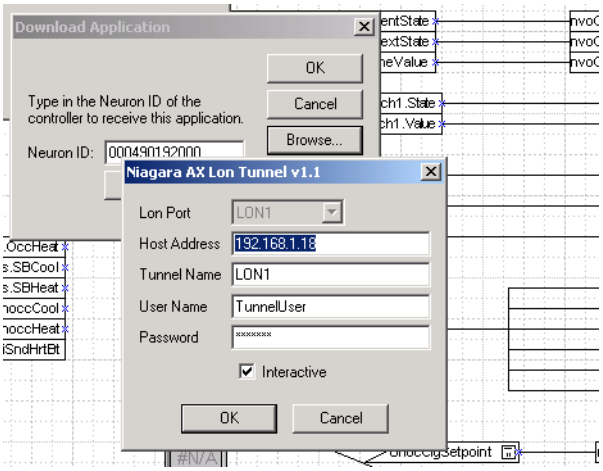
Lon tunneling usage notes

In addition to these notes, be sure to see "Best security practices for tunneling" on page B-5.

Lon tunneling is *not supported for usage with remote network management tools*, where the station is configured as "just a LON node" (see "Notes when configuring as Lon node" on page 3-40). In this case, the PC performing Lonworks network management should attach directly on the network.

When initiating a connection through the Lon tunnel, client-side usage is transparent except for the "virtual LON*n*" port used, and if "Interactive" is left enabled (typical) the resulting dialog right before the connection is attempted. Figure B-6 shows an example connection being initiated from a vendor's Windows LON-based application. Tunneling client messages appear if the connection fails.

Figure B-6 *Example vendor-specific Windows LON application initiating tunnel connection*



Speed of the tunnel connection may be slower that a direct (FTT-10) connection, due to the overhead of wrapping and unwrapping LonTalk messages in Niagara Fox and TCP/IP protocols. Tunneling is not recommended if using a "dialup modem" for connection to the target JACE station.

Tunneling client messages

When using a tunnel client, the specified user must be "authenticated" by the station before a connection is granted (established). If the user is not found, or if the entered password is incorrect, a popup message appears on the client PC (Figure B-7). Note the User Name and Password are both case-sensitive.

Figure B-7 *Authentication issue*

Caution *As previously cautioned, note that "basic authentication" is used in any client connection to the station's TunnelService, with possible security consequences. See "Best security practices for tunneling" on page B-5.*

Currently, only one tunnel connection is allowed per LonTunnel (or SerialTunnel). If another client application attempts a connection to that tunnel, a popup message appears on that PC (Figure B-8).

Figure B-8 *Tunnel busy*

Appendix C

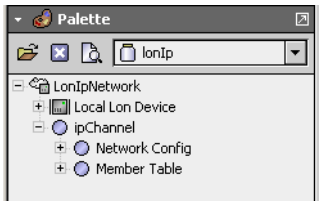
Lon over IP

JACE support was added for "Lon over IP" using a `lonIp` module, used in conjunction with the standard `lonworks` module. Starting in AX-3.5, support was extended, as described in these main sections:

Overview

The `lonIp` module became available starting with AX-3.2, and contains a palette with a *modified* **`LonNetwork`**— the default name is "`LonIpNetwork`". Figure C-1 shows it expanded in the palette.

Figure C-1 *Palette for lonIp module (AX-3.5)*



The modified LonNetwork communicates on an IP Lonworks *channel* as defined in CEA-852. In this document, this CEA-852 channel is referred to as the "LonIp channel". When using the LonIp driver, the station presents itself as an "*IP channel node*" member (as opposed to a Lon/IP *router* member).

CEA-852 specifies a "Configuration Server" (*Config Server*) that handles the distribution of routing information to all members of a LonIp channel.

- Prior to AX-3.5, the LonIp driver required a pre-configured, third-party Config Server already installed, along with one or more CEA-852 routers on the Lon Ip channel. The driver would not work otherwise.
- Alternatively, starting in AX-3.5, you can configure the JACE station to *operate as the Config Server* on the network of configured CEA-852 routers (Lon Ip channel). Or, you can use the previous method of referencing a pre-configured, third-party Config Server.

Note: *In either case, the LonIp channel must be* properly configured*—that is, all channel members must be recognized by the Config Server, and have up-to-date routing tables. Each channel member must be assigned a unique subnet/node address, but use the same channel ID.*

Note LonIp channel setup is outside the scope of normal NiagaraAX Lonworks station configuration, with the sole exception of the subnet/node address assigned to the station for use as an IP channel node. For example, setup of Lon/IP router members typically requires proprietary, vendor-specific configuration. Consult each device's documentation for details on how to set its Lonworks subnet/node address, etc.

Once the IP channel is *properly configured*, and the station has been added as an IP channel node, all other Lonworks functions will perform the same as if the JACE was connected directly to a LON trunk.

Also see "LonIp licensing", "LonIp architecture", and "LonIp FAQs".

LonIp licensing

A JACE must be properly licensed for LonIp, including separate feature entries for both "`lonworks`" and "`lonIp`" in its license. Note that device limits or proxy point limits for either feature may exist in a license.

If the JACE is not licensed with the `lonIp` feature, a **`LonIpNetwork`** may remain in a fault condition.

LonIp architecture

The **`LonIpNetwork`** is used in place of a standard **`LonNetwork`**. To use, you copy (drag and drop) this network from the `lonIp` palette into your station's **`Drivers`** folder. Note that compared to a standard LonNetwork, the LonIpNetwork has additional frozen "ipChannel" container slot with other child components (and a view) described in sections ahead.

LonIp FAQs

The following frequently asked questions apply to the LonIp driver.

Q: Is a physical LON adapter (FTT-10, etc.) required by the JACE to use the LonIp driver?

A: No LON adapter is required. The LonIp driver uses the JACE's Ethernet port—unlike the standard Lonworks driver, which binds any LonNetwork to a specific LON adapter by device name (`LON1` or `LON2` for example). In the LonIp driver, this device name is fixed, read-only as "`LonIp`".

See this in property sheet of a **`LonIpNetwork`**, by expanding its **`Lon Comm Config`** container.

Q: Why can't I add the LonIpNetwork from the Driver Manager, as with most other drivers?

A: Because there is no separate "LonIpNetwork" class. The LonIpNetwork palette entry is just "BLonNetwork" with the comm stack and the local device changed to support communications using IP.

Q: What is the purpose of the Config Server?

A: The Config Server collects address information from each of the members of the LonIp channel, and then updates all the other members with the collected information. It is necessary to have one central device responsible to collect and disseminate this member information, since the protocol does not provide a mechanism for members to discover each other..

Q: Why during a device discover do I get a lot of warning messages, similar to below?

```
WARNING [12:58:48 31-Jul-09 EDT][lonip] Received response message with no matching trans-
action.16 00 01 19 0f 00 04 01 00 00 00 00 00 00 00 00 21 03 00 00 08 26 81 80 00 01 01 01 01 9a 21
```

A: This is normal behavior for the LonIp driver (LonIpNetwork), as the driver receives and processes all messages on the wire addressed to our device. The discover process involves broadcasting a request/response message, and then processing the first response which closes out the transact. Other responses may be received after matching transact has completed.

When using the standard Lonworks driver (LonNetwork), these warning messages do not appear because the Neuron processes the messages off the wire, and it filters out all except the first response.

LonIp Setup

Use one of two setup methods, depending on how the CEA-852 Config Server is implemented.

- Setup on a LonIp channel with a third-party Config Server
- Setup on a LonIp channel with Niagara as the Config Server

Setup on a LonIp channel with a third-party Config Server

Use this method when the JACE is installed on a network that includes a configured third-party Config Server and configured CEA-852 routers.

To setup Lon over IP with a third-party Config Server

With the station opened in Workbench:

Step 1 Open the `lonIp` palette in the palette side bar.

Drag or copy a "LonIpNetwork" into the station's **`Drivers`** container.

Step 2 Double-click the LonIpNetwork for its **`Lon Device Manager`** view.

Then double-click its **`Local Lon Device`** to edit its **`Channel Id`**, **`Subnet`**, and **`Node`** address to be consistent with the LonIp channel the station will join as a member. **`Save`** when done.

Note: *Channel Id and subnet must match other devices on the channel, and the node address must be unique for that subnet.*

Step 3 Open the property sheet of the network's **`Ip Channel`**, **`Network Config`** component.

To do this from the Nav tree, expand the **`LonIpNetwork`**, then its **`ipChannel`**, then double-click **`Network Config`**.

Enter the IP address of the Config Server in the **`Config Server Ip`** property field, and then **`Save`** when done.

Note: *If behind a NAT (Network Address Translation) router, other* **`Network Config`** *properties need editing. See "Setup notes if a NAT router firewall" on page C-4.*

Step 4 At the Config Server, add a member to the IP channel for the JACE station. Refer to the vendor's documentation for the Config Server for this step.

Once contact is made with the Config Server, and the station becomes a member of the channel, the **`Member Table`** under the network's **`ipChannel`** will become dynamically populated.

From this point forward, other Lonworks functions will perform as if directly connected to a Lon trunk, that is, as if using the standard LonNetwork.

Note: *To see channel members in the table, double-click the network's* **`ipChannel`** *for its* **`Member Manager`** *view. Unlike in the other LonIp setup method (where Niagara operates as the Config Server), you do* not *need to edit or add any members in this view.*

Setup on a LonIp channel with Niagara as the Config Server

Use this method when the JACE is installed on a network that has configured CEA-852 routers, but there is no Config Server.

To setup Lon over IP with Niagara as the Config Server

With the station opened in Workbench:

Step 1 Open the `lonIp` palette in the palette side bar.

Drag or copy a "LonIpNetwork" into the station's **`Drivers`** container.

Step 2 Double-click the LonIpNetwork for its **`Lon Device Manager`** view.

Then double-click its **`Local Lon Device`** to edit its **`Channel Id`**, **`Subnet`**, and **`Node`** address to be consistent with the LonIp channel the station will join as a member. **`Save`** when done.

Note: *Channel Id and subnet must match other devices on the channel, and the node address must be unique for that subnet.*

Step 3 Open the property sheet of the network's **`Ip Channel`**, **`Network Config`** component.

To do this from the Nav tree, expand the **`LonIpNetwork`**, then its **`ipChannel`**, then double-click **`Network Config`**.

Change the property **`Is Config Server`** from `false` to `true`, then click **`Save`**.

Note: *After you* **Save**, *and then* refresh the property sheet, *the property "Config Server Ip" should no longer appear in the property sheet.*

Step 4 Double-click the network's **ipChannel** for its **Member Manager** view.

Use the **New** button to add a member for each CEA-852 router, and also one for the JACE station too. For each member, edit the device's IP address, IP port, and if needed, NAT IP address.

- In the member that represents the station, enter the JACE's own IP address. Typically, if using defaults, enter its Name as "Niagara" and leave the other properties (ipPort and natIpAddress) as is.
- For any member behind a NAT router, its **ipAddress** is the address on the *local* network, and the **natIpAddress** is the NAT router address on the *Internet* side.

All other properties (starting with "routerType") will be set with data obtained from each device.

Note: *For more details about setup behind a NAT router, see the next section "Setup notes if a NAT router firewall". For additional details on ChannelMembers, see "About ChannelMembers" on page C-5*

Setup notes if a NAT router firewall

If behind a NAT (Network Address Translation) router, in addition to other set up:

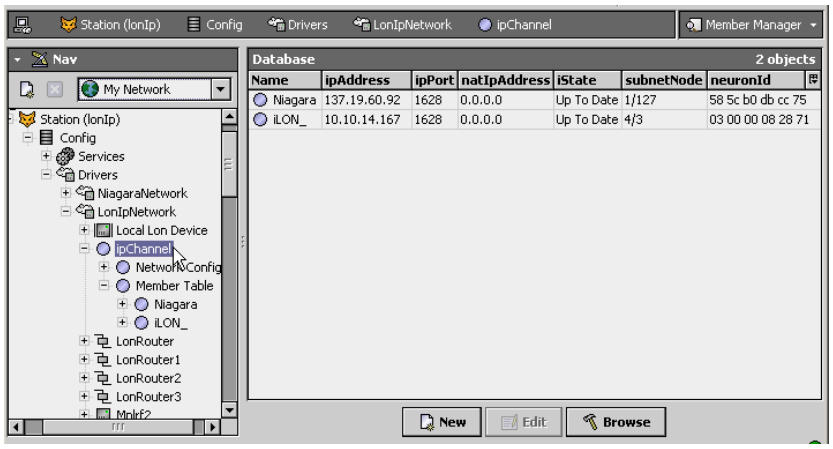
1. In the LonIpNetwork's ipChannel, Network Config property sheet, set the following properties:
   - **Use Extended Nat**
     Set this to `true`.
     Note: *This is automatically forced to true if extended NAT messages are detected.*
   - **Nat Ip Address**
     Enter the IP address of the *Internet side* of the NAT router.
2. The NAT router must be set up to forward any port used by the station as a LonIp channel node, as well as any other LonIp channel members.
   If the station is the Config Server, then the Config Server port must also be forwarded. Refer to manufacturer instructions for setting up port forwarding.

Note: *Setup of LonIp routers for use behind a NAT router requires setting the NAT router address and possibly other config information that is not defined in CEA-852. Therefore, refer to each device manufacturer's instructions for such configuration. Also, note that setup may require use of their Config Server.*

About the IpLonNetwork's Member Table

Starting in AX-3.5, a new **Member Manager** view was added in the LonIp driver, as the default view of the LonIpNetwork's **ipChannel** component (double-click **ipChannel** to see this view). This view provides a manager view on the LonIp channel's **Member Table**.

Figure C-2 *Example Member Manager view*



- If LonIp is set up with a third-party (non-Niagara) Config Server, you can use this for "informational access" to the Member Table. This table is composed of automatically-created child "ChannelMember" components. No additions or edits using this view are necessary.
- If LonIp is set up with Niagara (this station) acting as the Config Server, use this view to add and edit the necessary **ChannelMember** components, via the **New** and **Edit** buttons. See "Setup on a LonIp channel with Niagara as the Config Server" on page C-3 for a procedure.

See the next section "About ChannelMembers" for more details.

About ChannelMembers

ChannelMember components typically represent Lon/IP 852 routers, as well as one required entry for the JACE station itself (as a "channel node"). If using a third-party (non-Niagara) Config Server, all ChannelMembers are automatically added once communications to the CS have been established.

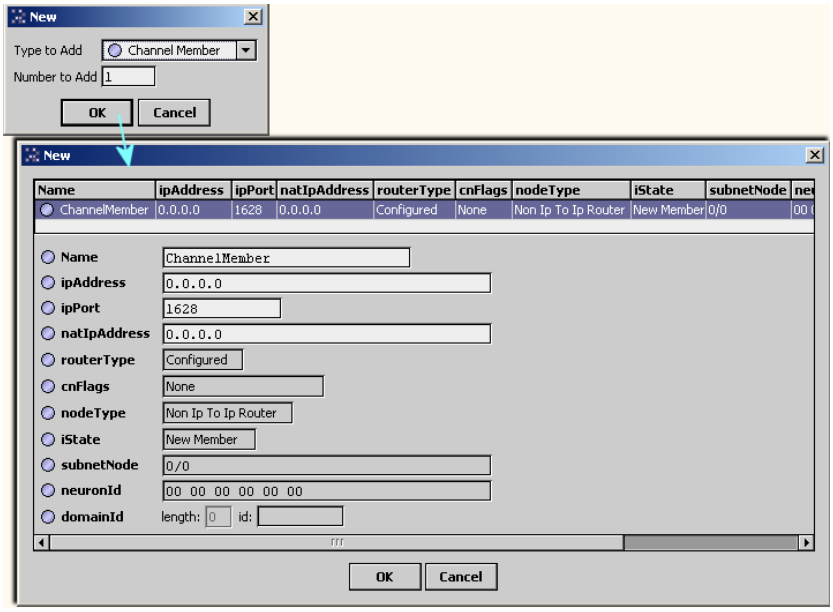
If configuring Niagara as the CEA-852 Config Server, you must manually add ChannelMembers. Once added, you can monitor these component's status in the **`Member Manager`** view, and if necessary, also edit and open a browser connection to any selected one (for possible configuration, if the device provides a built-in web server). ChannelMembers also provide a number of right-click actions.

See the following sections for more details:

New (and Edit) dialogs

Adding a new ChannelMember in the Member Manager view provides dialogs shown in Figure C-3.

Figure C-3 *New ChannelMember dialogs*



Only the first four properties are configurable in either the **`New`** or **`Edit`** dialog, described as follows:

- **Name**
  Name of the NiagaraAX component to represent this device, it is also the value of slot "cnName" of the created component. Default value is "`ChannelMember`".
  For the component that represents *this station*, you typically enter "`Niagara`" to match the default value of the "Net Name" property of the ipChannel's **`Network Config`** component.
- **ipAddress**
  IPv4 address of the device on the LAN. Enter the unique IP address of the device.
  For the component that represents *this station*, enter the host JACE platform's IP address.
- **ipPort**
  Software port monitored for messages from the Config Server, with the default (and typically used) conventional value of `1628`.
- **natIpAddress**
  If behind a NAT router, the device's assigned external (Internet) address, if applicable.
  If not behind a NAT router, leave at the default `0.0.0.0` value.

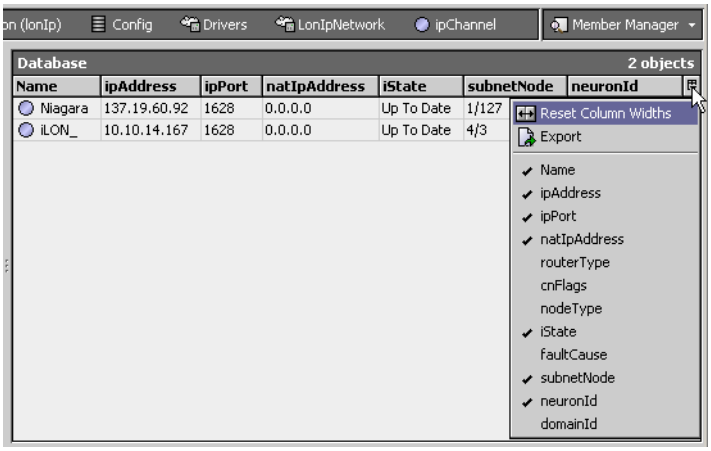
ChannelMember read-only properties Remaining property values are obtained from each device, and are also available for display in columns in the Member Manager view, as follows:

- **routerType**
  Enumerated descriptor for the router type. (Not shown as default column in the view)
  - Configured

  - Learning
  - Bridge
  - Repeater
- **cnFlags**
  Enumerated descriptor for the channel flag. (Not shown as default column in the view)
  - None
  - All Broad Casts
  - Security
  - All Broad Casts _Security
- **nodeType**
  Enumerated descriptor for the channel member type. (Not shown as default column in the view)
  - Uninitialized
  - Non Ip to Ip Router
  - Ip Channel Node
  - Ip Channel Proxy
  - Ip to Ip Router
- **iState**
  Enumerated descriptor for member status, as either:
  - New Member
  - Sent D R Req
  - Sent D R
  - Sent C R Req
  - Up to Date
  - Error
- **subnet/Node**
  Assigned Lonworks subnet/node address, unique on site for this device.
- **neuronId**
  Device's unique Lonworks Neuron ID.
- **domainId**
  Assigned Lonworks working domain. (Not shown as default column in the view)

Note the read-only items above are available for data columns in the Member Manager view, as shown in Figure C-4 (only default items shown selected).

Figure C-4 *Table selector in Member Manager view of ipChannel component*



A "faultCause" property is also available for display in the **`Member Manager`**, for example to display a possible issue such as "`Device will not ack DEVICE_CONFIGURATION`".

Member Manager buttons

If the Niagara station is set up as the CEA-852 Config Server, use the **`New`** button in the Member Manager to add a member for each CEA-852 router and also one for the JACE station too. See the section "Setup on a LonIp channel with Niagara as the Config Server" on page C-3.

The **`Edit`** button provides a similar dialog for ChannelMembers already in the station database.

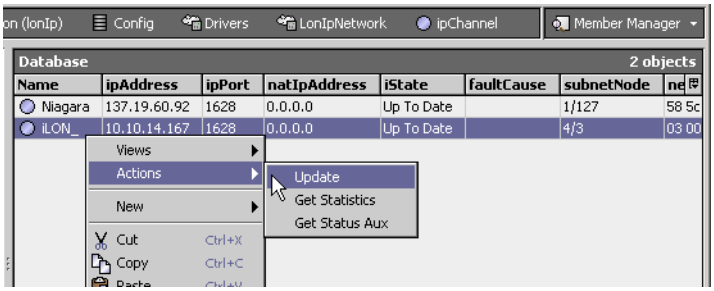
See "New (and Edit) dialogs" on page C-5 for more property details.

Use the **Browse** button on any selected ChannelMember to launch a separate client web browser window to that device's local IP address. This is simply a convenience feature. Often, Lon/IP 852 routers include a web server interface that allows (proprietary) remote configuration of the device.

ChannelMember actions

Any selected ChannelMember has an action menu, as shown in Figure C-5.

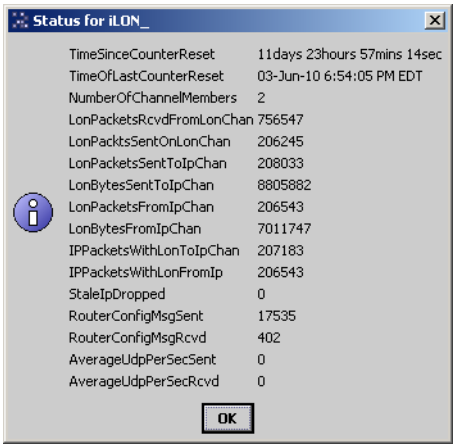
Figure C-5 *Action menu for any selected IpLon ChannelMember*



Actions may be useful when troubleshooting the LonIp driver, and are described as follows:

- **Update**
  Forces an update of LonIp channel information to that device. Note this also occurs automatically upon station startup, and whenever LonIp channel information changes.
- **Get Statistics**
  Produces a popup dialog with various statistics for the selected channel member (Figure C-6).

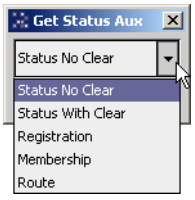
Figure C-6 *Statistics popup dialog from ChannelMember's action Get Statistics*



This information may be particularly useful when troubleshooting.

- **Get Status Aux**
  Produces a popup action submenu for the selected channel member (Figure C-7).

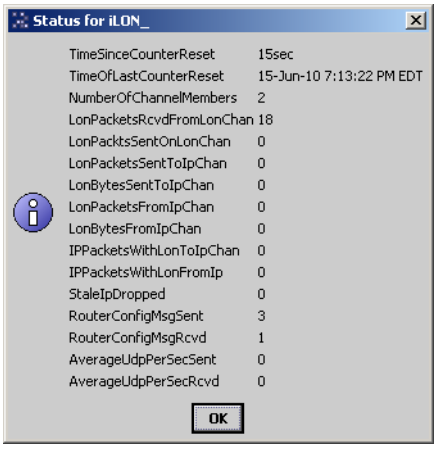
Figure C-7 *Popup submenu action dialog from ChannelMember's action Get Status Aux*



Get Status Aux submenu actions include the following:

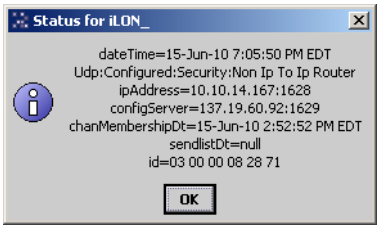
- **Status No Clear** — Reads current status without resetting the statistics counters. Equivalent to the **Get Statistics** action, with resultant popup previously shown in Figure C-6.
- **Status With Clear** — Reads current status and resets (clears) the statistics counters. The resultant popup shows many "0" entries, as shown below in Figure C-8.

Figure C-8 *Status popup dialog after Status With Clear action (from Get Status Aux)*



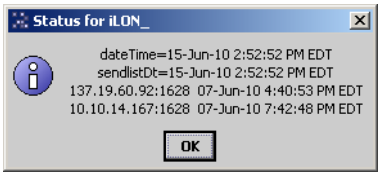
- **`Registration`** — View this member's registration information in a popup (Figure C-9).

Figure C-9 *Status popup dialog from ChannelMember action Get Status Aux, Registration*



- **`Membership`** — View this member's internal membership list in a popup (Figure C-10).

Figure C-10 *Status popup dialog from ChannelMember action Get Status Aux, Membership*



- **`Route`** — View this member's routing information in a popup (Figure C-11).

Figure C-11 *Status popup dialog from ChannelMember action Get Status Aux, Route*