Technical Document

Niagara^{AX} Sedona Framework TXS 1.2 Networks Guide

February 25, 2013

Niagara^{AX} Sedona Framework TXS 1.2 Networks Guide

Confidentiality Notice

Trademark Notice

Copyright and Patent Notice

Contents

PREFACE

Preface

This document explains NiagaraAX-3.7 and later integration of Sedona Framework devices, using Sedona Framework TXS 1.2. This document is also available in the Niagara Workbench help system after installing the released Sedona Framework TXS 1.2 bundle, which includes `docSedonaNetworks.jar`.

This preface has the following sections:

About this document

As noted earlier in this Preface, this document applies to NiagaraAX integration of Sedona Framework devices in AX-3.7 and later using Sedona Framework TXS 1.2, and has the following main sections:

- Sedona Framework Network Driver Installation
  Explains the NiagaraAX Workbench requirements, including its "Sedona environment" setup. Also covered are processes required to install the necessary Sedona-related NiagaraAX modules in remote (JACE) hosts, as well as configuring the remote "Sedona environment" in a JACE. Related procedures require *platform connections* to remote JACEs.
- Sedona Network Quick Start
  Provides several step-by-step procedures for getting started with a Sedona Framework network (either SedonaNetwork or SedonaJen6lpNetwork), working online with a *station connection*.
- Niagara Sedona Framework Concepts
  Provides concepts behind both types of Sedona Framework networks, including most components and views. Includes reference information on almost all properties and actions of components, from networks to devices to proxy points and action points.
- Sedona Framework Plugin Guides
  Provides brief summaries of the various Sedona Framework-related views, with links back to the more detailed concepts sections. Entries are used in NiagaraAX Workbench context-sensitive help "On View".
- Sedona Framework Component Guides
  Provides brief summaries of the different Sedona Framework-related components, with links back to the more detailed concepts sections. Entries are used in NiagaraAX Workbench context-sensitive help "Guide On Target".
- Appendix A - Sedona Framework Tools for Workbench
  Provides overview information about various tools added to NiagaraAX Workbench when enabled for Sedona Framework TXS, along with links to other documents with more details.
- Appendix B - Sedona environment management
  Explains managing Sedona environment files on a remote NiagaraAX host (typically a JACE) using a platform connection from Workbench, via the **`Sedona Environment Manager`** view. Starting in Sedona TXS-1.2, this feature allows Sedona device provisioning and app access "through the JACE", and enables complete job backup using station saves and backups.
  Related procedures are in the "Sedona Framework Network Driver Installation" section of this guide.
- Appendix C - Sedona users and roles management
  Explains the optional central management of Sedona users in networked Sedona devices, via the station's **`SedonaUsersManager`** service, along with the **`Sedona Device User Manager`** view of any Sedona network. Starting in Sedona TXS-1.2, this feature allows central coordination of Se-

dona users in networked devices, as well as a method to globally change the "super user" in all devices on a Sedona network.

Sedona Framework network FAQs

Below are some frequently asked questions (FAQs) about NiagaraAX integration of Sedona Framework devices.

Q: **What is the difference between Sedona Framework TXS and Sedona Framework?**

A: Sedona Framework TXS (Tridium eXtended Services) represents all of the value added, commercial components of the Sedona Framework that Tridium has created above and beyond the Open Source Sedona Framework release. Sedona Framework TXS 1.2 is based on the NiagaraAX-3.7 release and the Open Source Sedona Framework 1.2 release.

Q: **What changed in Sedona Framework TXS 1.2 since Sedona Framework TXS 1.1?**

A: A number of enhancements have been made in the NiagaraAX integration of Sedona Framework devices, including host-level (JACE station) support for provisioning and app access of networked Sedona devices, time synchronization from NiagaraAX to Sedona devices, a centralized method to manage Sedona user accounts, and various offline engineering methods and tools. For more details refer to the *Sedona TXS 1.2 Feature List* in the Knowledge Base area of niagara-central.com.

Q: **Will Sedona networks using Sedona TXS 1.2 support Sedona Framework devices based on earlier versions of the Sedona Framework, such as Sedona Framework 1.0?**

A: Yes, the Sedona network drivers in Sedona TXS 1.2 will support networking of both 1.0-based devices as well as 1.2-based devices. Support includes the ability of station-based provisioning and app access (Sedona Tools and Sox Gateway, respectively) for all networked devices, along with time synchronization and user management from NiagaraAX. However, note online "device discovery" for the Ethernet/WiFi-based SedonaNetwork requires devices based on Sedona Framework 1.2.

A: Further, note that Sedona Framework devices are based on either Sedona Framework 1.0, or Sedona Framework 1.2 (there is no Sedona Framework 1.1). Sedona Framework TXS 1.1, the previous Sedona Framework TXS release, is compatible with Sedona Framework 1.0 devices only.

Sedona Framework terms

The following is a list of terms and abbreviations used in this document when describing networks of Sedona Framework devices. For general NiagaraAX terms, see the Glossary in the *User Guide*. Note that this glossary may grow over time, or may else simply be eliminated.

6LoWPAN Acronym for IPv6 over Low power Wireless Personal Area Networks. It is an international open standard that enables using 802.15.4 and IP together. Sedona was created with 6LoWPAN networking capability in mind, reflected in its DASP and Sox protocols.

app The app in a Sedona Framework device is its application of Sedona components and services, including links between them, plus all configuration properties. Components and services are selected from kits installed in the device. In this way, an app is analogous to a station on a NiagaraAX platform. Using a Sox connection, Workbench can be used to engineer an app and save it to the device's Flash memory. Starting in Sedona TXS-1.2, each networked Sedona device also has an "App Manager" view under its "Sedona Tools" in a station connection, for getting/putting a device's app in the station's file space. This feature requires use of the **`Sedona Environment Manager`** (platform view) to station's host.

A Sedona Framework device may also have a "fallback app", residing in a different area of Flash memory. A fallback app is started via some local device-specific method, for example, pressing push-buttons in combination.

bundle Sedona Framework TXS software is distributed in a "bundle". A bundle is a special image (set of files) that is available from Niagara-Central (www.nigara-central.com). You use the **`Sedona Installer`** tool in Workbench to install a bundle. Bundles include Sedona TXS-specific NiagaraAX modules (`.jar` files), among other items. For more details, refer to the *NiagaraAX Sedona Installer Guide*.

Chopan Or CHoPAN, for Compressed HTTP over Personal Area Networks. It is a Tridium proprietary protocol used in Sedona Framework TXS to support Jennic-based devices, particularly battery-powered devices. Chopan runs over UDP/IP and is a "session-less" protocol (unlike Sox), offering a number of advantages over Sox in certain applications. For related details, see "About the Chopan Virtual gateway" on page 3-40.

coordinator In a network of Jennic-based devices, the JACE station acts as the single "coordinator" node for all child nodes, using the Sedona Jennic option card installed in that JACE. The coordinator maintains info about its child nodes, each of which may provide routing functionality or be end devices. Properties of the station's SedonaJen6lpNetwork configure the coordinator's operating parameters. See "SedonaJen6lpNetwork coordinator properties" on page 3-29.

Alternatively, a special Sedona Jennic "USB stick" can be used as the coordinator with Sedona Framework Workbench (or Sedona Framework TXS-enabled Niagara Workbench) to support a Sox connection to a *single* Jennic-based device. No network functionality is possible. Setup of this USB coordinator is done using the "New Jennic Wireless Adapter" tool in Workbench. See the *Jennic Serial Tools Guide* for details.

DASP For Datagram Authenticated Session Protocol. This is the low-level, secure session-based protocol that Sox utilizes. DASP operates in networks that include 6LoWPAN and resource-limited devices. Sedona Framework network and device components in NiagaraAX provide debug properties that allow examining DASP and Sox messaging.

hibernating device Refers to a type of Jennic-based device that is typically powered by an onboard battery or batteries. Such a device invariably "hibernates" (sleeps) the majority of time, periodically "waking up" for short periods to execute routines and exchange data with other devices. Such devices require configuration using CHoPAN (Chopan). See "Hibernating devices" on page 3-24.

Currently, Sedona Framework support for hibernating devices is not widely available. However, the SedonaJen6lpNetwork driver in the NiagaraAX station is "ready" for such device support if this changes. Other sections of this document that mention hibernating devices also note this.

Jennic-based A Jennic-based device is the term used in Tridium tech docs for a wireless Sedona Framework device based on a Jennic micro-controller, with built-in 802.15.4 connectivity and 6LoWPAN stack support. Such devices are modeled as "SedonaJen6lpDevices" in the NiagaraAX station of a JACE controller (with an installed "Sedona Jennic" option card), under a SedonaJen6lpNetwork. See "About Jennic-based devices" on page 3-24.

JenNet The Jennic protocol that manages wireless 802.15.4 network formation and message routing, sitting above the 802.15.4 layer and below the 6LoWPAN layer. JenNet provides a "self healing tree" network, versus a mesh network. In a SedonaJen6lpNetwork, the JACE station is always the top "coordinator" node of the network tree. A special diagnostic "Pan Sheet" view of the network provides JenNet data.

kit Sedona kits are the basic unit of modularity of Sedona software, encapsulating code, types, and metadata. A kit is analogous to a module on a NiagaraAX platform. The app in a device instantiates components and services contained in its installed kits. You must have the appropriate kits available on your Workbench platform to change a device's "core" software. Starting in Sedona TXS-1.2, kit files are also among the "Sedona environment files" you also install in JACE, using the **`Sedona Environment Manager`** (platform) view. This allows Sedona device provisioning of networked devices through a station connection, using a "Kit Manager" view under the "Sedona Tools" of each networked device.

manifest Each kit has a corresponding manifest, with all metadata needed by tools like Workbench to Sox connect to a Sedona Framework device, and for a station to support Sedona proxy points in that device. Manifest files are compact XML files, named using a *`kitName-checksum`* convention similar to kit files, but with an `.xml` extension.

Starting in Sedona TXS-1.2, manifest files are also among the "Sedona environment files" you also install in JACE, using the Sedona Environment Manager (platform) view. This allows Sedona device provisioning of networked devices through a station connection, as well as "Sox Gateway" access to the app in networked devices.

You can also use the Manifest Manager view of either Sedona Framework network type to manage kit manifests on the NiagaraAX host (JACE or Supervisor), or the Sedona Manifest Manager view in Workbench tools to manage manifests on your Workbench host. However, starting in Sedona TXS-1.2 the primary use for the Manifest Manager view is for updating kit manifests in the local (Workbench) environment. For complete details, refer to the *Sedona Manifest Manager - Engineering Notes* document.

PAN Personal Area Network, a generic term for a device network that is typically limited to a small area.

Sedona Framework device A Sedona Framework device runs a Sedona Framework app in a Sedona Framework VM (virtual machine), using installed Sedona Framework kits, and is configured in Workbench using a Sox connection. Sedona Framework devices may vary in a number of ways, including device connectivity—for example Ethernet/IP, WiFi, or 802.15.4 (wireless PAN).

Sox Sox is the standard protocol used to communicate with Sedona Framework devices. It runs over UDP via the lower-level DASP protocol. Workbench always uses Sox to connect to (open) a Sedona Framework device. A Niagara station also uses Sox to discover and (typically) read and write to Sedona proxy points. However, if a Jennic-based device is configured with Chopan, that comm type can be used for proxy point updates.

WPAN Wireless Personal Area Network, a PAN (personal area network) using a wireless technology.

Document change log

Updates (changes/additions) to this *Niagara^{AX} Sedona Framework TXS 1.2 Networks Guide* document are listed below.

- Updated: February 25, 2013
  Document updated as a "version split" for Sedona TXS 1.2, with many changed and new sections.
- Publication: November 10, 2011
  Initial document for Sedona TXS 1.1.

Chapter 1

Sedona Framework Network Driver Installation

Prior to creating a Sedona network in an AX-3.7 or later station using Sedona Framework TXS 1.2, there are requirements for configuring your Workbench installation (including its "Sedona environment"). You then use a platform connection to install the needed Sedona-related NiagaraAX modules in the station's host platform, as well as any required license file changes. Additionally, starting in Sedona TXS-1.2 you need platform access to install relevant "Sedona environment files" in the remote (JACE) host.

Such configuration enables new features provided by Sedona TXS-1.2, including "Sedona Tools" provisioning and "Sox Gateway" access of networked Sedona devices, via a normal station (Fox) connection from Workbench.

This section explains the necessary platform configuration processes, and contains the following subsections:

Sedona network type review

There are currently two different NiagaraAX Sedona Framework network (or driver) types, depending on communications method between the NiagaraAX station and the Sedona Framework devices:

- **SedonaNetwork**
  For Sedona Framework devices that provide an Ethernet port (or WiFi) that utilizes TCP/IP. This Sedona Framework TXS-1.2 driver is capable of running on any AX-3.7 or later platform, meaning any model JACE or a Supervisor. The NiagaraAX host license requires the `sedonanet` feature.
- **SedonaJen6lpNetwork**
  For wireless "Jennic-based" devices, which use 6LoWPAN over 802.15.4. This Sedona Framework TXS-1.2 driver requires a JACE-2, -6, -7 or JACE-x02 XPR (with installed Sedona Jennic option card) as host platform, running AX-3.7 or later. This driver is licensed separately from the other (SedonaNetwork) one, where the JACE host license to have the `jen6lp` feature.

If needed, and if properly licensed, a JACE station can have *both* types of Sedona Framework networks. Installation of the Sedona modules and files described in this section apply to both network types.

Workbench and Sedona file requirements

Before upgrading a JACE from a previous revision, or before installing software modules in a JACE, use the Workbench **`Sedona Installer`** tool to install the latest Sedona Framework TXS 1.2 bundle for Workbench, and restart Workbench. This ensures the latest Sedona Framework-related modules are available in Workbench for installation in remote JACE controllers. For related details, see the *NiagaraAX Sedona Installer Guide* document.

After enabling your Workbench for Sedona Framework TXS 1.2, you should also import other Sedona files in your "local" Sedona environment (your Workbench PC) for the specific types of Sedona Framework devices to be networked in NiagaraAX stations. Obtain these Sedona files from the Sedona Framework device vendor(s) for those devices. Such files include Sedona kit "manifest" files (`.xml`), Sedona kit files (`.kit`), and Sedona platform archive files (`.par`).

Device vendors may furnish such files in a zipped format, or as individual files. Either way, you use the Workbench **`Sedona Installer`** tool to import these files in your local (Workbench) Sedona environment, by choosing the option to "Import Sedona environment files". For more details, see the section "Importing Sedona environment files" in the *NiagaraAX Sedona Installer Guide*.

JACE workflow

The following lists the high-level steps required to configure an AX-3.7 or later JACE with the Sedona TXS-1.2 environment, such that the JACE's station supports "Sedona Tools" provisioning of networked devices, as well as "Sox Gateway" access to the Sedona app in networked devices.

1. A prerequisite is that you have AX-3.7 or later Workbench, from which you have previously installed a Sedona TXS-1.2 bundle, using the "Sedona Installer" tool. This host should be licensed for features `sedonaProvisioning` and `sox` (and if running a station with a Sedona Network, also `sedonanet`).

   Note the JACE also requires all these license feature entries, and possibly others; for example, the `jen6lp` feature if it has a SedonaJen6lpNetwork. See "Sedona Framework licensed features" in the *NiagaraAX Sedona Installer Guide* for further details.
2. Also, your Workbench host PC should have (in its local Sedona environment) those Sedona kit files, manifest files, and platform database files that are needed to support the provisioning of Sedona devices that are networked under the target JACE—or that will be. You will be transferring (copying) those files to the JACE's Sedona environment.
3. Using Niagara Workbench, open a platform connection to the JACE and install (or upgrade) these Sedona TXS-1.2 modules:
   - nsedona
   - sedona
   - sedonac
   - sedonanet
   - sedonaProvisioning (note this module was not used by a JACE before Sedona TXS-1.2)

   If the JACE has a Sedona Jennic option card installed (for wireless Jennic-based Sedona devices), install or upgrade these additional modules:
   - jen6lp
   - pansheet
   - platJen6lp

   Use either the **`Software Manager`** view to install these modules, or else use the platform **`Commissioning Wizard`**, if other platform commissioning is also needed (say to install an updated license file, and/or upgrading the JACE's release level to AX-3.7). The wizard's "Select modules" step *always runs*, where you can select needed software modules, such as those listed above. For details, see "About the Commissioning Wizard" in the *NiagaraAX JACE Install and Startup Guide*.
4. After commissioning the JACE with the Sedona TXS-1.2 modules listed above, open (or if necessary reopen) a *platform connection* to it.

   Now use the platform **`Sedona Environment Manager`** tool to install the necessary files in its Sedona environment. See "Installing Sedona environment files in a JACE". *See NOTES below!*

   - A JACE-2 series (NPM2-based) controller requires a "maxHeap" license feature (for expanded 128MB memory) to fully support an installed Sedona environment. If you have a JACE-2 without this license feature, contact your product vendor to determine how to add this requirement.
   - When installing Sedona environment files on a JACE, be mindful that each transferred file and folder counts towards its "file handle" allocation limit (also used by history files). In AX-3.7, file handle limits by JACE platform types are:
     – JACE-2: series: 1000
     – JACE-4/5 or JACE-6/6E series: 2000
     – JACE-7 series: 3000
     – JACE-NXT, JACE-NXS, or AX SoftJACE: no limit

     Typically, this is not an issue if you transfer over only those environment files required by the Sedona devices (platforms) that are networked under the JACE. However, it is recommended that you do not simply transfer over your entire Niagara Workbench's Sedona environment for the reason above—particularly if that environment is quite large, containing platforms and kits that are not applicable.
5. Edit the **`BackupService`** in the JACE station to allow Sedona device backup zip files to be included in *station backups*. See "Station backup notes" on page B-6.

Installing Sedona environment files in a JACE

This section provides quick steps on using the **Sedona Environment Manager** platform view, when platform-connected to a AX-3.7 or later JACE with Sedona Framework TXS-1.2 modules.

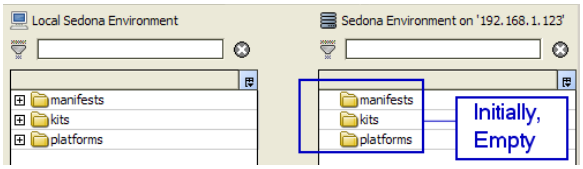
Note: *This is a late step in a process that includes prerequisites. See "JACE workflow" on page 1-2, including bulleted NOTES in process step 4. For the rationale behind using this platform view, see "Sedona environment management" on page B-1.*

To install Sedona environment files to a remote host

Perform the following from your Sedona TXS-1.2-enabled Niagara Workbench PC or Supervisor:

Step 1 Open a platform connection to the JACE.

Step 2 Double-click the Sedona Environment Manager to open this platform view.



Initially, the right-side of the view shows the three folders on the JACE *empty*—you can tell because there is no "expand" ⊞ control to the left of each folder.

Step 3 In your Local Sedona environment (*left-side*), click to expand folders to see hierarchical subfolders with file items of interest. Click items to toggle between marked and unmarked for transfer, as:

manifest file marked or manifest file unmarked.

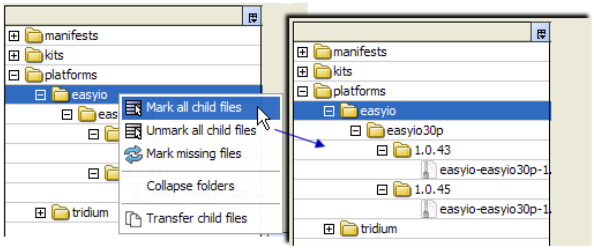
kit file marked or kit file unmarked.

platform file marked or platform file unmarked.

Note while any file is marked, the "transfer to remote" button becomes enabled:

You can also mark (select) *entire folders* of files to transfer, at any hierarchical level.

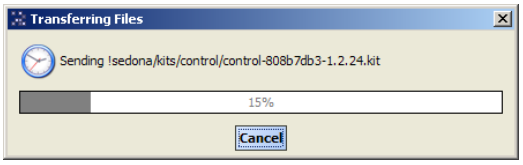
To do this, right-click any folder for a popup menu.



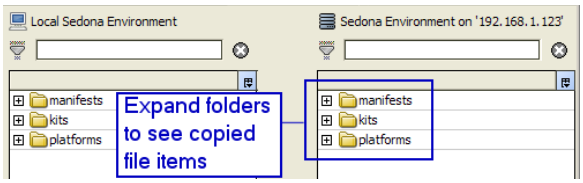
Click **Mark all child files** to select all, or **Unmark all child files** to deselect all.

Other menu options work as titled (**Mark missing files** is most useful *after* transferring files).

Step 4 When done marking files in your Local Sedona environment, click (transfer to remote).
A popup **Transferring Files** dialog shows progress and details on files being sent (copied).



The transfer finishes with a brief **`Loading Sedona Environment`** popup, where after the remote Sedona environment tree (right-side) appears with expandable folders.



Expand folders on the remote host's (right-side) Sedona environment to see copied file items.

Step 5 Repeat, as needed, to select additional files to transfer from your local Sedona environment to the remote JACE's Sedona environment. You can do as many transfers as needed.

Note: *JACE controllers have a finite limit of file handles. In general, it is best not to transfer over your entire local (Workbench) Sedona environment, but only those files needed by the networked Sedona device types.*

Other controls in the **`Sedona Environment Manager`** can also be useful. For example, a *few* are described as follows:

- To filter/find specific manifests or kits (on either side), click in the filter box at top , and type in characters that must be included in file names for them to display, for example: `se` would limit to manifests and kits folders for `pulse` and `serial`, and possibly others. You could also use this for a "checksum" string portion, for example: `8d` or `6a6`
  Click the clear button at the end of the filter box to remove the filter (all folders display).
- To delete specific files (on either side), click one or more to select, then click the delete button at the *bottom* of that side. A popup confirmation dialog appears—if you click OK, file(s) are deleted.

Caution *Be careful with delete, particularly on your Local Sedona environment side! There is no undo.*

For details on all aspects of this platform view, see "Sedona Environment Manager" on page B-3.

Also, we recommend you adjust your station's BackupService to support Sedona device backups done from networked devices' "Sedona Tools", which get written to the station's file space. See "Editing the station's BackupService" on page B-7 for a procedure.

Chapter 2

Sedona Network Quick Start

This section provides a collection of procedures to get started with either NiagaraAX Sedona Framework driver in a typical online scenario, that is with a station (Fox) connection from Workbench. Like other NiagaraAX drivers, you can do most configuration from special "manager" views and property sheets using Workbench.

Note: *For best results, before starting the "station building" procedures described in this section, make sure you have completed tasks in the previous "Sedona Framework Network Driver Installation" section. This requires platform connections to remote JACE hosts.*

These are the main quick start subsections:

- Configure a SedonaNetwork
  - Add a SedonaNetwork
  - Discover and add SedonaDevices (devices must be based on Sedona 1.2 to support discovery)
  - Manually add SedonaDevices
- Configure a SedonaJen6lpNetwork
  - Add a SedonaJen6lpNetwork
  - Configure key SedonaJen6lpNetwork properties
  - Discover and add SedonaJen6lpDevices
  - Rename SedonaJen6lpDevices
- Optional in most cases: Run the Manifest Manager to ensure kit manifests are loaded
- Create Sedona proxy points (and action points)
- Configure to allow Sox tunneling

Note: *A separate engineering notes document, "*Using the Sedona Jennic option card*", also provides basic steps to get started with the* **`SedonaJen6lpNetwork`** *driver for wireless Jennic-based devices, and includes additional overview and background information.*

Configure a SedonaNetwork

The SedonaNetwork (for Ethernet and WiFi Sedona Framework devices) is the simplest of the two types of Sedona Framework networks. To configure a SedonaNetwork, perform the following main tasks:

- Add a SedonaNetwork
- Manually add SedonaDevices

Add a SedonaNetwork

Note: *The host requires certain modules and licensing. See "JACE workflow" on page 1-2.*

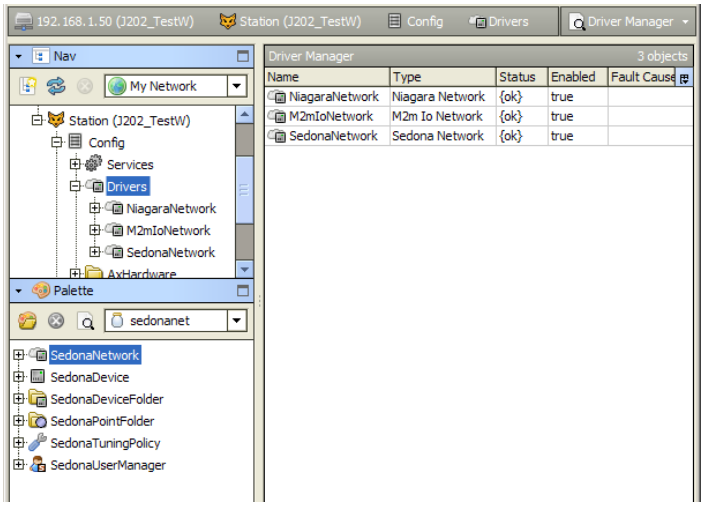
Add a SedonaNetwork

With the station open in Workbench:

Step 1 In the Nav tree, expand the station's **`Config`** node to reveal the **`Drivers`** container.

Step 2 Open the **`sedonanet`** palette in your Workbench palette side bar.

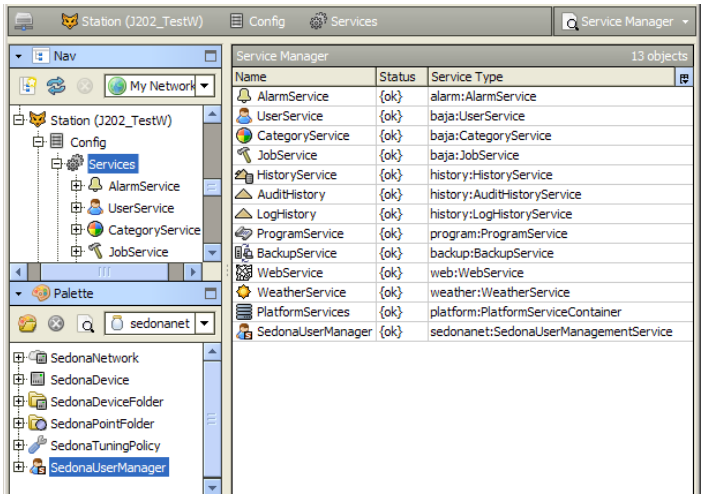
Step 3 From the **sedonanet** palette, *drag* (or copy and paste) a **SedonaNetwork** into the station's **Drivers** container. In the popup **Name** dialog, either rename the network, or just use the default name.



A SedonaNetwork named "SedonaNetwork" (or whatever else), is under your Drivers folder. Leave properties at defaults for now. For details, see "About the SedonaNetwork component" on page 3-6.

Step 4 In the Nav tree, expand the station's **Services** container.

Step 5 (Optional) From the **sedonanet** palette, *drag* (or copy and paste) the **SedonaUserManager** into the station's **Services** container. In the popup **Name** dialog, either rename or use the default name.



A SedonaUserManagementService named "SedonaUserManager" (or whatever else), is under your Services container. Leave properties and child components at defaults for now. For related details, see "Sedona users and roles management" on page C-1.

Discover and add SedonaDevices

Ethernet/IP or WiFi-equipped Sedona Framework devices that are based on Sedona TXS-1.2 may support device discovery. If so, and they are on the same LAN as the station's host, you can use the learn/discovery mode of the Sedona Device Manager to discover and add SedonaDevices to represent them.

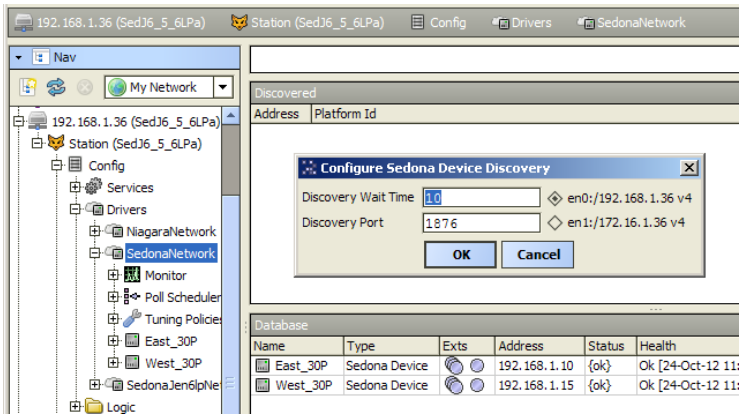
Note: *You must* manually add *any Ethernet or WiFi-based Sedona devices that do not support device discovery. This includes* any device *based on Sedona Framework 1.0. See "Manually add SedonaDevices" on page 2-5.*

Discovering and adding SedonaDevices

Note: *Device discovery uses UDP/DASP multicasting on the selected Ethernet/IPv4 adapter of the station's host platform (IPv6 is not supported in Sedona TXS-1.2). The host's TCP/IP (IPv4) settings must be configured correctly, including subnet mask and default gateway, for online discovery to work.*

Step 1 (Optional) Use the **New Folder** feature to make one or more device folders, each with its own device manager view—a common method to "group" devices. Then double-click a folder as in Step 2.

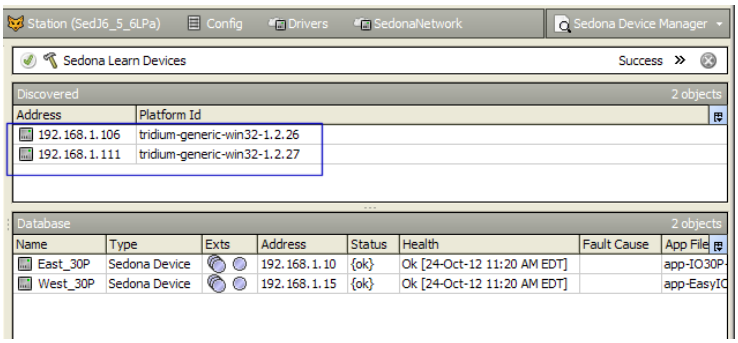
Step 2 Double-click the SedonaNetwork for its default Device Manager view, and click the **Discover** button. A two-pane "learn mode" appears, with a **Configure Sedona Device Discovery** popup.



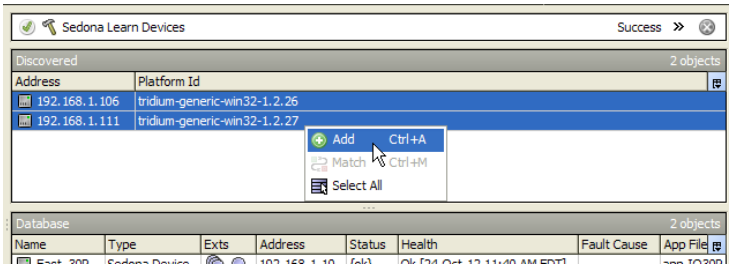
Step 3 In the **Configure Sedona Device Discovery** dialog, select the Ethernet/IP port to use, and either accept or change the defaults for discovery wait time (10 seconds) and Sox port (1876) as shown above. A JACE controller with both LAN ports enabled lists "en0" (LAN1) and "en1" (LAN2).

Click **OK** to begin the discovery. A "Sedona Learn Devices" job launches, with progress bar at the view top.

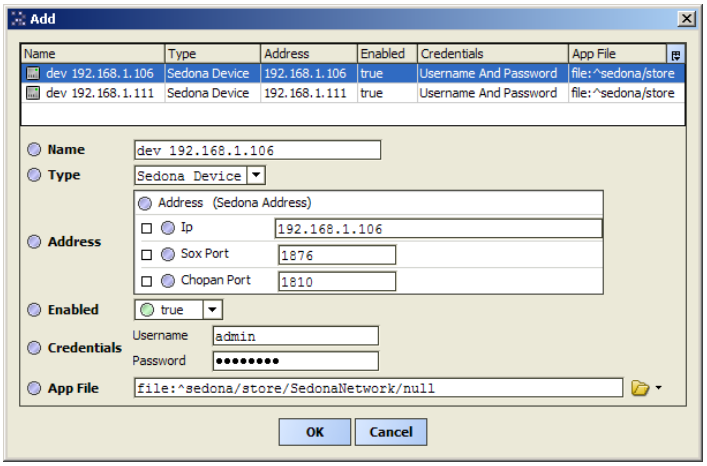
Step 4 Discovered devices appear in the *top* **Discovered** pane following job completion.



Click to select discovered devices (hold Ctrl while clicking to select multiple) and click **Add**.



Step 5 The **Add** dialog appears for the selected device(s).



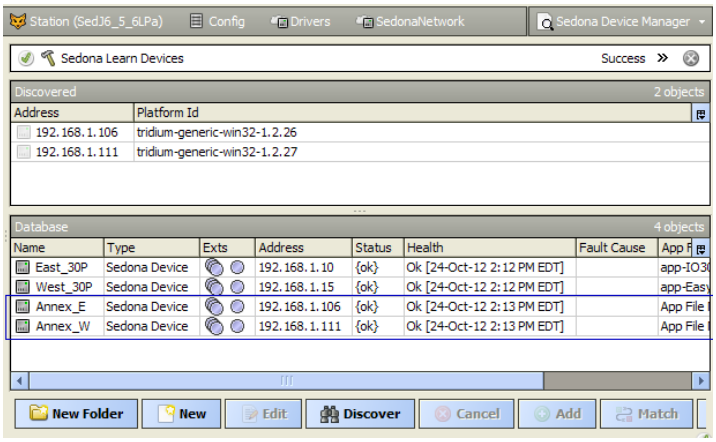
For each row (device) in the top of the Add dialog, do as follows:

- In the **Name** field, you typically enter another name instead of "`dev IpAddress`" (may be changed later at any time). This is a Niagara name—it does not need to match any Sedona Framework app or component name.
- In the **Credentials** area, enter Sedona user credentials for a Sox connection.
  - Typically you leave **Username** at default `admin` (for this standard user).
  - In **Password** field, enter the required password.
- For discovered devices leave all other fields at default values, described below.
  - **Sedona Address** area:
    - **Ip** field reflects the IPv4 address of the Sedona Framework device (e.g. `192.168.1.4`).
    - **Sox Port** reflects the Sox port used of the Sedona Framework device (default 1876).
    - *Disregard* **Chopan Port** — currently, Chopan applies to SedonaJen6lpDevices only.
  - Leave **Enabled** at `true`.
  - **App File** is initially null (`file:^sedona/store/SedonaNetwork/null`); it remains so until the device is "associated" with an app file (one way is to "get" the app currently in the device using the **Sedona Tools** under the networked device). For details, see "Sedona device Association" on page 3-20.

Click **OK** to add the SedonaDevice(s) to the network.

If successful, you should see each device with "ok" health in the Sedona Device Manager (Figure 2-1).

Figure 2-1 *Newly-added SedonaDevices*



Step 6 Repeat the discover and device add if necessary for additional TXS-1.2 based Sedona Framework devices.

Other device component properties and containers are available on each device's property sheet. For related details, see "About the SedonaDevice component" on page 3-12.

Each device has a points extension, with a default **Sedona Point Manager** view. However, before adding proxy points you should (at a minimum) make sure the station's host (typically a JACE) has the necessary Sedona kit manifest files.

Note: *Even better, and required to support "Sedona device provisioning through the station", you should install all the necessary "Sedona environment files" on that station's host. In addition to the Sedona kit manifest files, this includes all the kit files and platform archive files used by networked Sedona devices.*

For quick start procedures, see:

- "Installing Sedona environment files in a JACE" on page 1-3 (recommended over loading just kit manifests).
- "Run the Manifest Manager to ensure kit manifests are loaded" on page 2-12. (not necessary if installing Sedona environment files).
- "Create Sedona proxy points (and action points)" on page 2-13.

Manually add SedonaDevices

Ethernet/IP or WiFi-equipped Sedona Framework devices based on Sedona Framework 1.0 do not support device discovery; you need to manually add SedonaDevice components to represent them.

Note: *Devices based on Sedona Framework 1.2 may support device discovery, such that manual addition is unnecessary. See "Discover and add SedonaDevices" on page 2-2.*

Add SedonaDevices

Note: *You need to know the IP addresses of the Sedona Framework devices. The IP address for each device should each be unique and static, and reachable by the Niagara station.*

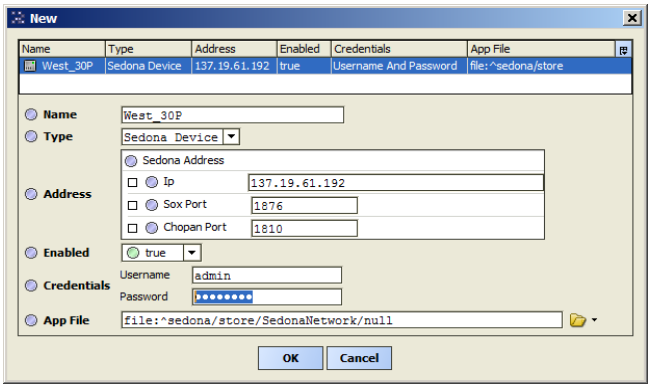
Step 1 Double-click the SedonaNetwork for its **`Sedona Device Manager`** view.

Step 2 In the Sedona Device Manager, click the **`New`** button to add a new Sedona Device.

(Optional) Use the **`New Folder`** feature to make one or more device folders, each with its own device manager view—a common method to "group" devices. Then double-click a folder and click **`New`**.

Click **`OK`** for the **`New`** dialog for adding a device (Figure 2-2).

Figure 2-2 *New dialog for adding a SedonaDevice*

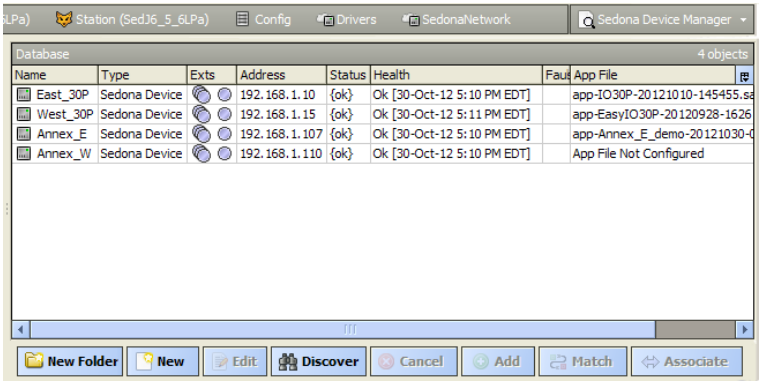


- In the **`Name`** field, you typically enter another name instead of "SedonaDevice" (may be changed later at any time). This is a Niagara name—it does not need to match any Sedona Framework app or component name.
- In the **`Sedona Address`** area:
  - In the **`Ip`** field, enter the known IPv4 address of the device (for example: `192.168.1.4`).
  - Leave **`Sox Port`** at default 1876 (unless the device's Sedona Framework app uses a different Sox port).
  - *Disregard* **`Chopan Port`** — currently, Chopan applies to SedonaJen6lpDevices only.
- Leave **`Enabled`** at `true`.
- In the **`Credentials`** area, enter Sedona user credentials for a Sox connection.
  - Typically you leave **`Username`** at default `admin` (for this standard user).
  - In **`Password`** field, enter the required password.
- **`App File`** is initially null (`file:^sedona/store/SedonaNetwork/null`); it remains so until the device is "associated" with an app file (one way is to "get" the app currently in the device using the **`Sedona Tools`** under the networked device). For related details, see "Sedona device Association" on page 3-20.

Click **`OK`** to add the SedonaDevice to the network.

If successful, you should see the device with "ok" Health in the Sedona Device Manager (Figure 2-3).

Figure 2-3 *Newly-added SedonaDevices*



Step 3 Repeat Step 2 to manually add other Ethernet and/or WiFi Sedona Framework devices. As with other drivers, you can add multiple devices in a single **New** dialog—in this case, click on the row for each device to edit properties like Name, Ip, etc.

Other device component properties and containers are available on each device's property sheet. For related details, see "About the SedonaDevice component" on page 3-12.

Each device has a points extension, with a default **Sedona Point Manager** view. However, before adding proxy points you should (at a minimum) make sure the station's host (typically a JACE) has the necessary Sedona kit manifest files.

Note: *Even better, and required to support "Sedona device provisioning through the station", you should install all the necessary "Sedona environment files" on that station's host. In addition to the Sedona kit manifest files, this includes all the kit files and platform archive files used by networked Sedona devices.*

For quick start procedures, see:

- "Installing Sedona environment files in a JACE" on page 1-3 (recommended over just installing the kit manifests).
- "Run the Manifest Manager to ensure kit manifests are loaded" on page 2-12 (not necessary if Sedona environment files have been installed in the JACE).
- "Create Sedona proxy points (and action points)" on page 2-13.

Configure a SedonaJen6lpNetwork

The SedonaJen6lpNetwork is a "variation" of a SedonaNetwork, having all its features—plus additional ones to support 802.15.4 wireless (Jennic) operation, and other items for support of possible battery-powered (hibernating) devices.

To configure a SedonaJen6lpNetwork, perform the following main tasks:

- Add a SedonaJen6lpNetwork
- Configure key SedonaJen6lpNetwork properties
- Discover and add SedonaJen6lpDevices
- Manually add SedonaDevices

Add a SedonaJen6lpNetwork

Note: *The JACE host controller requires an installed Sedona Jennic option card, as well as certain modules and licensing. See "JACE workflow" on page 1-2.*

Add a SedonaJen6lpNetwork

With the station open in Workbench:

Step 1 In the Nav tree, expand the station's **Config** node to reveal the **Drivers** container.

Step 2 Open the **jen6lp** palette in your Workbench palette side bar.

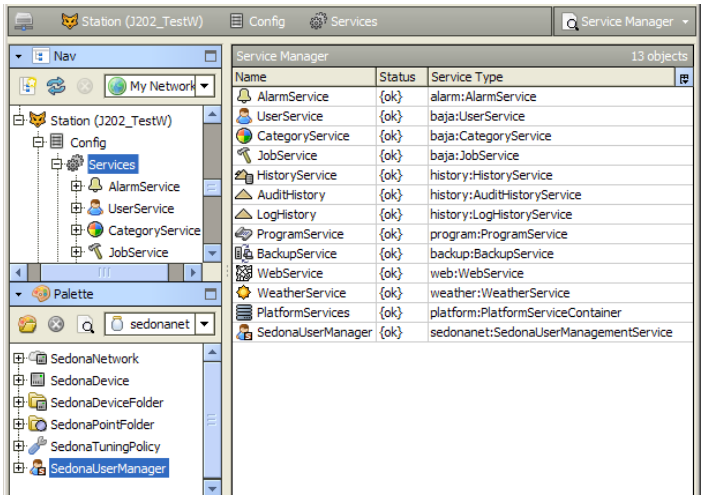
Step 3 From the **jen6lp** palette, *drag* (or copy and paste) a **SedonaJen6lpNetwork** into the station's **Drivers** container. In the popup **Name** dialog, you can rename the network—or, simply use the default.

A SedonaJen6lpNetwork named "SedonaJen6lpNetwork" (or whatever you named it), is under your Drivers container.

Step 4 Open the **sedonanet** palette in your Workbench palette side bar.

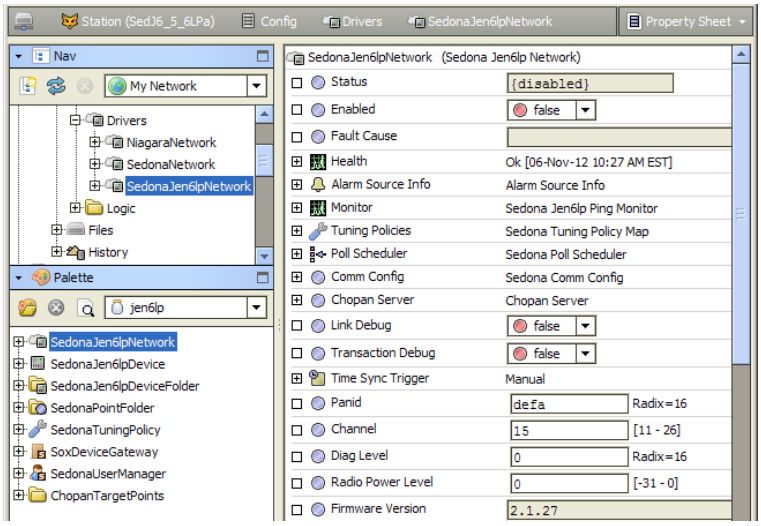
Step 5 In the Nav tree, expand the station's **Services** container.

Step 6 (Optional) From the **sedonanet** palette, *drag* (or copy and paste) the **SedonaUserManager** into the station's **Services** container. In the popup **Name** dialog, either rename or use the default name.



A SedonaUserManagementService named "SedonaUserManager" (or whatever else), is under your Services container. Leave properties and child components at defaults for now. For related details, see "Sedona users and roles management" on page C-1.

Step 7 Go to the property sheet of the newly-added SedonaJen6lpNetwork.



As copied from the palette, the SedonaJen6lpNetwork is disabled, and requires entered values in several configuration properties before it can be operational.

See "Configure key SedonaJen6lpNetwork properties" on page 2-7.

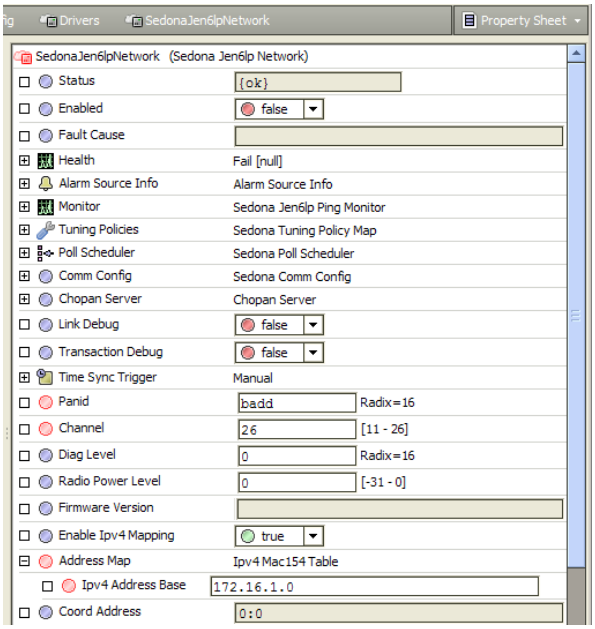
Configure key SedonaJen6lpNetwork properties

Do this after adding the SedonaJen6lpNetwork (see "Add a SedonaJen6lpNetwork" on page 2-6).

Note: *You must know the Jennic PANID and channel map range used by the wireless Jennic-based devices, in order to configure the Sedona Jennic option card in the JACE to operate as the coordinator for them.*

Configuring key SedonaJen6lpNetwork properties

Step 1 Open the property sheet for the **SedonaJen6lpNetwork**, if not already open

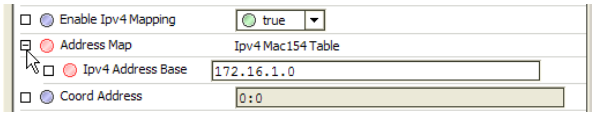


Step 2 Set the **Panid** and **Channel** used by the wireless network of Jennic-based devices. A few properties of this network are described as follows:

- **Enabled**
  Initially false (disabled). *After* configuring and saving other properties in remaining steps, you change it tor true and save, to initiate all changes.
- **Panid**
  Jennic PAN ID (Personal Area Network identifier), in hexadecimal, range 0x0000 to 0xFFFE for the network. The JACE station acts as the coordinator for this network. Panid should match the PANID in use by installed devices (check with the vendor of installed devices). The default Panid is: `defa`
- **Channel**
  RF 2.4GHz channel number to use, from 11 to 26. This channel must be included in the "channel map" of installed devices.
- **Address Map**
  Container to specify the IPv4 address network "base" as well as hold entries for discovered nodes. See the next step.

For more details on properties above, see "SedonaJen6lpNetwork coordinator properties" on page 3-29.

Step 3 Expand the **Address Map**.



In the **Ipv4 Address Base** field, either accept the default address base network (`192.168.1.0`), or enter another private IPv4 subnet for the JACE to map discovered wireless Jennic-based devices.

Note: *The IPv4 address base must be on a different subnet than the subnet used by the JACE for normal IP communications. For further details, see the* **Address Map** *description in the section "SedonaJen6lpNetwork coordinator properties" on page 3-29.*

The **Ipv4 Address Base** subnet must fall within the Class A, B, or C address range, as follows:

- 10.0.0.0 to 10.255.255.0 (Class A)
- 172.16.0.0 to 172.31.255.0 (Class B)
- 192.168.0.0 to 192.168.255.0 (Class C)

As new wireless devices are discovered, they are assigned the next available IP address. The first address (`n.n.n.0`) is reserved for the coordinator.

For example, with the base address at default (`192.168.1.0`), the coordinator is `192.168.1.0`, and the first device to associate with the coordinator is assigned `192.168.1.1`, the next device `192.168.1.2`, and so on. These devices automatically appear as dynamic entries under the network's Address Map.

Step 4 Save all changes made in the SedonaJen6lpNetwork property sheet.

Step 5 Make sure the **Enable Ipv4 Mapping** property is true, and set the network's **Enable** property to true, and **Save** again.

The status of the network should change from disabled to ok, and the "Coord Address" should change from `0:0` to an actual address (for example: `158d00:9b50b`). Under the **Address Map** in the property sheet, entries should begin to populate under the Ipv4 Address Base. Within a minute or two, all address map entries should be complete.

Note: *In the future, after changing any of the following network properties, you must disable then re-enable the SedonaJen6lpNetwork (or else restart the station) for them to be effective:*

- *Panid*
- *Channel*
- *Radio Power Level*

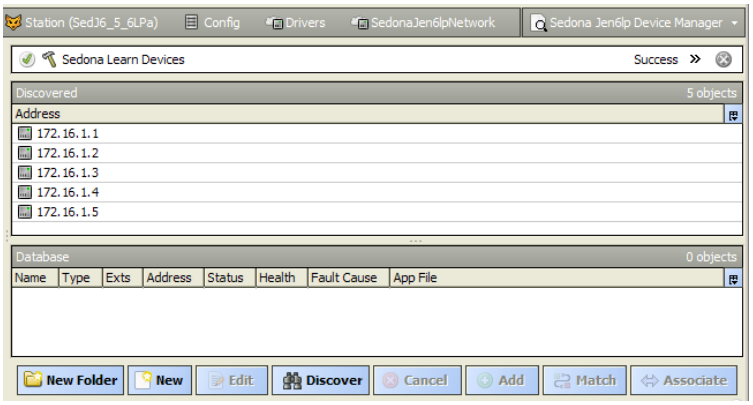
For more details on *key properties* of the network, see "SedonaJen6lpNetwork coordinator properties" on page 3-29. For reference information on all network properties and actions, see "SedonaJen6lpNetwork properties" on page 3-26 and "SedonaJen6lpNetwork actions" on page 3-32.

Discover and add SedonaJen6lpDevices

Do this after configuring key SedonaJen6lpNetwork properties (see "Configure key SedonaJen6lpNetwork properties" on page 2-7). SedonaJen6lpDevices represent wireless Jennic-based devices. For background details, see "About Jennic-based devices" on page 3-24.

Adding discovered Jennic-based devices

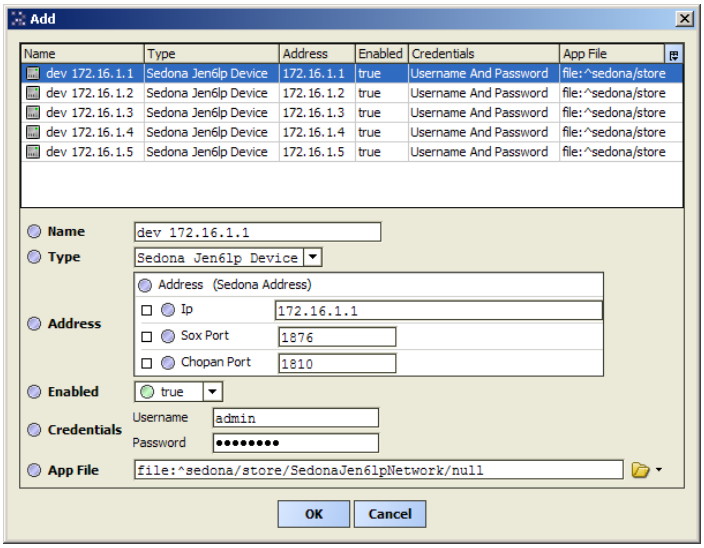
Step 1 Double-click the SedonaJen6lpNetwork for its default Device Manager view, and click the **Discover** button. A "Sedona Learn Devices" job quickly runs, and shows discovered devices in the top pane.



(Optional) Use the **New Folder** feature to make one or more device folders, each with its own device manager view—a common method to "group" devices. Then double-click a folder and do a discover.

Step 2 Click to select one or more devices, then click the **Add** button. Figure 2-4 shows the **Add** dialog.

Figure 2-4 *Example Add dialog when adding multiple devices, all with default entries*



Often properties can be left at defaults, including (initially) Name. After adding devices to the database you can rename devices with more meaningful text. See "Rename SedonaJen6lpDevices" on page 2-11.

Step 3 Click **OK** to add devices to the database.

Figure 2-5 *Example added Jennic-based devices with default names*

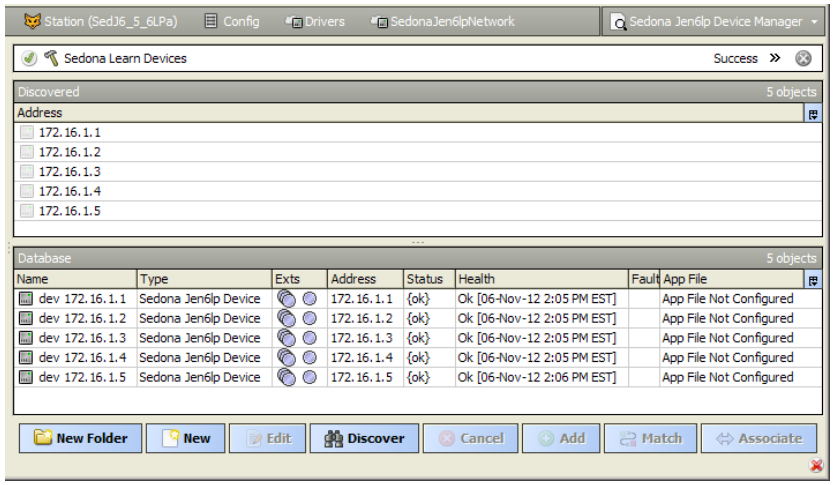


Figure 2-5 shows all discovered devices added to the station database using default names. Typically, now you rename all the SedonaJen6lpDevice components. See "Rename SedonaJen6lpDevices" on page 2-11.

Other device component properties and slots are available on each device's property sheet. For related details, see "About the SedonaJen6lpDevice component" on page 3-36.

Each device has a points extension, with a default **Sedona Point Manager** view. However, before adding proxy points you should (at a minimum) make sure the station's host (typically a JACE) has the necessary Sedona kit manifest files.

Note: *Even better, and required to support "Sedona device provisioning through the station", you should install all the necessary "Sedona environment files" on that station's host. In addition to the Sedona kit manifest files, this includes all the kit files and platform archive files used by the networked Sedona devices.*

For quick start procedures, see:

- "Installing Sedona environment files in a JACE" on page 1-3 (recommended over installing just the kit manifests).
- "Run the Manifest Manager to ensure kit manifests are loaded" on page 2-12 (not necessary if Sedona environment files have been installed in the JACE).
- "Create Sedona proxy points (and action points)" on page 2-13.

Rename SedonaJen6lpDevices

Do this after adding discovered devices (see "Discover and add SedonaJen6lpDevices" on page 2-9). Rename each device component according to its purpose and location. Note that wireless Jennic-based devices are initially mapped into the network's specified IPv4 Address Base (subnet) in an undefined order—e.g. you cannot "pre-specify" in the Sedona Framework app of a device for it to be "device 1" or "device 2" on a network.

To force an order, you *could* power on only *one* device at a time, add it in the station, and then rename the device component as known appropriate. Apart from that, there are a couple of possible ways to make the association between an added Jen6lp device component and a specific Jennic-based device:

- Service pin method
- Match address map entries to recorded MAC addresses

Note: *It is recommended that you also set the "Device Type" property for every added SedonaJen6lpDevice, working from the device's property sheet. Change from "Unknown" to either: "Router" or "End Device", and then* **Save**. *Only devices known to be incapable of router functionality should be set to "End Device". For related details, see "SedonaJen6lpDevice properties" on page 3-36.*

Service pin method

If the device's Sedona Framework app supports it, a "service pin" can be invoked from the physical device. For example, this may be possible from a device by pressing a pushbutton. This can be useful to make a positive identification from that device.

For related details, see "Service Pin support" in the *Sedona Framework Chopan Usage* document.

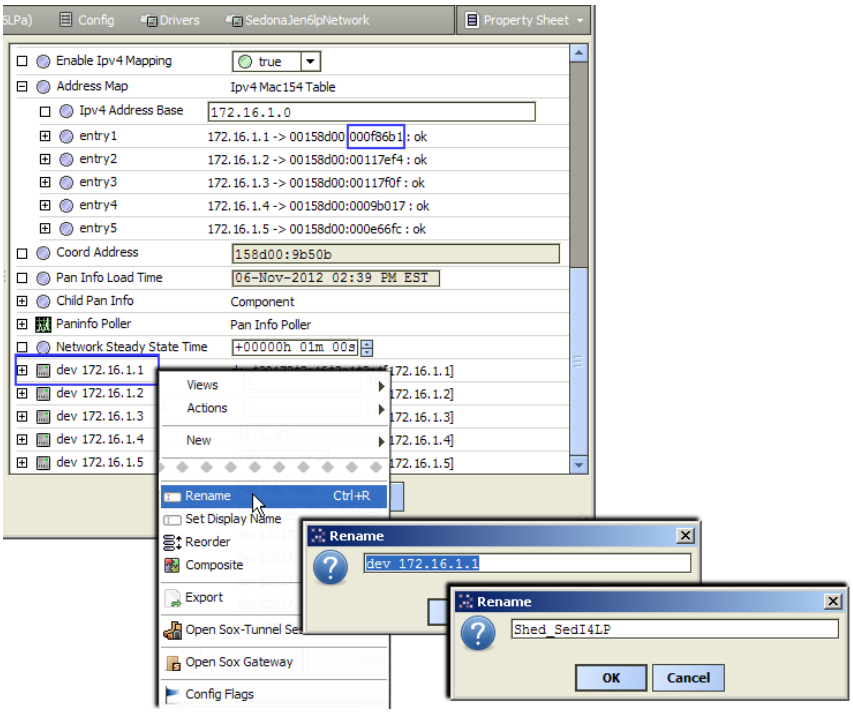
Match address map entries to recorded MAC addresses

This method always works, but it requires each device to have its unique MAC address on a printed label. It also requires careful note taking by the installer.

As you physically install the Jennic-based devices, make careful note of each one's unique MAC address, recording the last four bytes (eight characters) of its 64-bit address. For example, if you install a device with MAC address `00158d00:00117f0f`, record it as `00117f0f` in a list.

Compare your recorded list of devices against the SedonaJen6lpNetwork's "Address Map" entries (in the network's property sheet, expand the Address Map and note the mapped IPV4 addresses to device MAC addresses). See Figure 2-6.

Figure 2-6 *MAC address of Address Map entries used against recorded list, to help device renaming*



As shown in Figure 2-6, in the network's property sheet, below the **Address Map** area, you can right-click on each child device component, and rename it according to your notes.

Figure 2-7 *Five example Jen6lp devices added and renamed in station*

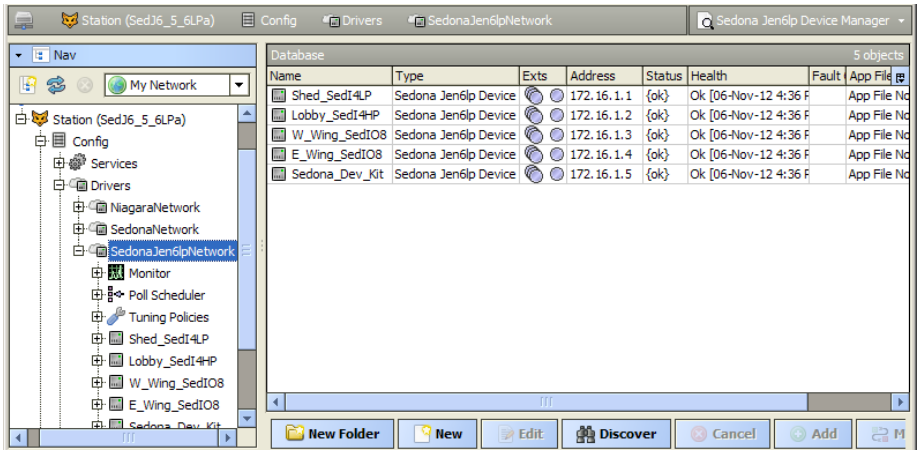


Figure 2-7 above shows five Jen6lp devices added and renamed in the station.

Run the Manifest Manager to ensure kit manifests are loaded

Note: *Starting in Sedona TXS-1.2, the importance of a Sedona network's* **`Manifest Manager`** *has diminished in favor of installing the complete set of "Sedona environment files" required by networked Sedona devices on the NiagaraAX platform hosting the station. This includes not only kit manifests, but also kits and platform files. This permits Sedona device provisioning "through the station", in addition to support for proxy points. Therefore, you can safely skip this section—and* instead install Sedona environment files.

For procedures, see sections "JACE workflow" on page 1-2 and "Installing Sedona environment files in a JACE" on page 1-3 in the "Sedona Framework Network Driver Installation" section.

To support Sedona proxy points, the NiagaraAX host (typically JACE) running the station with any Sedona Framework network requires all the Sedona "manifest" files for all the kits in each type of networked Sedona Framework device. These are the *same* manifest files required on your NiagaraAX Workbench host to support a Sox connection (or a *tunneled* Sox connection) to these devices.

Note: *The following quick start procedure assumes you already have all the necessary kit manifest files on your Sedona-enabled NiagaraAX Workbench host, located in the "*`manifests`*" subfolder of your working "*`sedona.home`*" (starting in AX-3.7,* `sedona.home` *is invariably* `!sedona`*). For the purposes of this procedure, consider this collection of manifest files your "Local Database".*

Complete details on Sedona manifests and using the **`Manifest Manager`** *are in the* Sedona Framework Manifest Manager - Engineering Notes *document.*

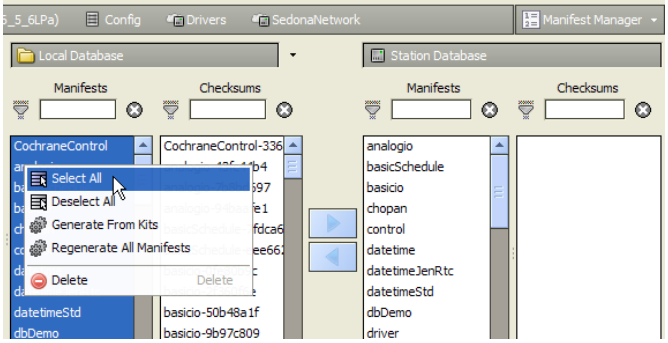
To run the Sedona Manifest Manager for the Sedona Framework network

This procedure applies to either Sedona Framework network type (SedonaNetwork or SedonaJen6lpNetwork), and results in the identical view and selections.

Step 1 With the station opened, right-click the **`SedonaNetwork`** or **`SedonaJen6lpNetwork`** and select: **`Views > Manifest Manager`**. This brings up the **`Manifest Manager`**.

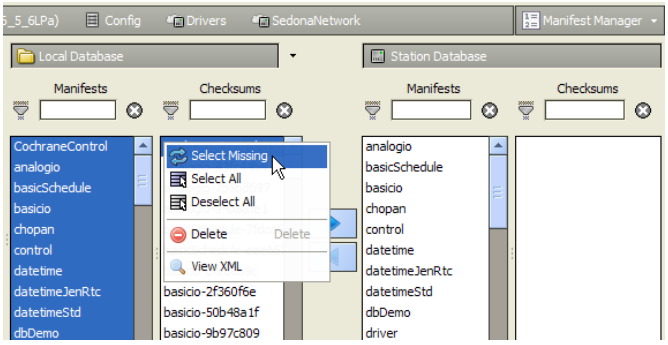
The *left side* of the view is your "**`Local Database`**"; the *right side* is the "**`Station Database`**".

Step 2 Right-click in the (far left) **`Manifests`** column of your **`Local Database`** and choose **`Select All`**.



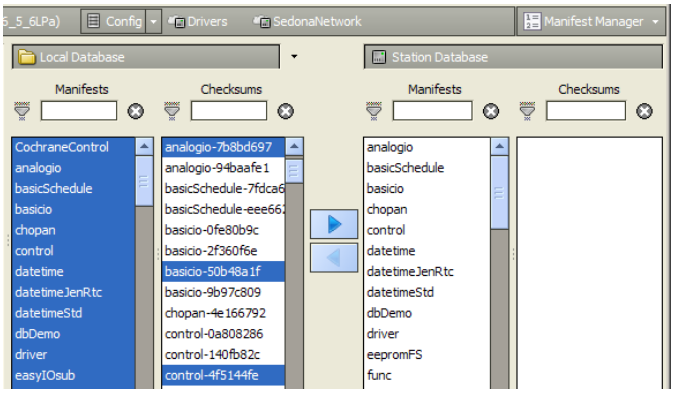
All kits in the Manifests column become highlighted, as shown above.

Step 3 Right-click in the **Checksums** column of your **Local Database**, and choose **Select Missing**.



This typically results in one or more **Local Database** "Checksums" entries becoming highlighted, which you can see by scrolling down that column, as shown below. In some cases it may result in *all entries selected* (usually this is OK, as these files are small and do not use much space).

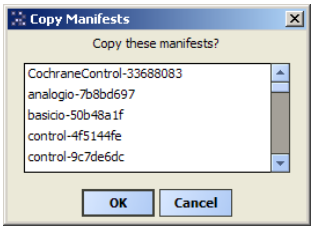
Figure 2-8 *Manifests in the Local Database but missing in the Station Database are highlighted*



The "Local Database" to "Station Database" manifest copy  button also becomes enabled.

Step 4 Click this copy button to copy the manifests to the station's host.

This produces a confirmation dialog listing the manifest files to be copied, as shown below.



Step 5 Click **OK** to copy the manifest files to the station's host.

The manifests are transferred and are now *immediately* available in the hosting station to support Sedona proxy point operations, such as online point discovery (no station restart required).

Create Sedona proxy points (and action points)

Note: *Before creating Sedona proxy points, you should make sure the station host (typically JACE) has the necessary Sedona kit "manifest" files for all the Sedona Framework devices. Otherwise, errors will occur when trying to discover points. Even better would be to install all appropriate "Sedona environment files" on the station's host platform, which includes kit files and platform files, in addition to kit manifest files.*

For procedures, see sections "JACE workflow" on page 1-2 and "Installing Sedona environment files in a JACE" on page 1-3 in the "Sedona Framework Network Driver Installation" section.

If Sedona device provisioning "through the station" is not needed, or in the case of a JACE-2 *without the "*`maxHeap`*" license feature for expanded 128MB memory, you can simply transfer only the manifest files. See "Run the Manifest Manager to ensure kit manifests are loaded" on page 2-12.*

As with device objects in other drivers, each Sedona Framework device (whether SedonaDevice or SedonaJen6lpDevice) has a **Points** extension that serves as the container for proxy points. The default view for any Points extension is the Point Manager (in this case, the **Sedona Point Manager**). You use it to discover and add Sedona proxy points and action points under any Sedona device component.

Note: *The* **Sedona Point Manager** *differs from other point managers in that actions on Sedona components are separately (and individually) modeled in the station as "Sedona action points", different from Sedona proxy points (standard Niagara control components with a Sedona proxyExt).*

You create both point types, that is proxy points and action points, from discovered slots of Sedona components in a device using the Sedona Point Manager.

To create Sedona proxy points and action points

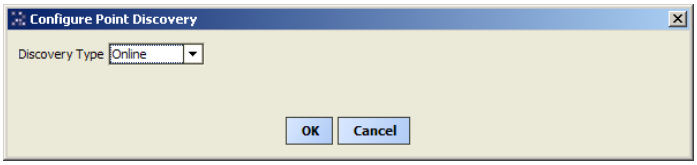
Once a Sedona device is added to the network (SedonaNetwork or SedonaJen6lpNetwork), you can discover the Sedona component slots in its app to select for modeling in the NiagaraAX station. Use the following procedure:

Step 1 In the **Sedona 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 **Sedona Point Manager**.

Step 2 (Optional) Use the **New Folder** feature to make one or more point folders, each with its own point manager view—a method to "group" proxy points under a device. Then double-click a folder to continue.

Step 3 Click **Discover** to initiate point discovery using Sox connectivity. The view goes to a split-pane (learn mode) and a **Configure Point Discovery** popup dialog appears, as shown below.

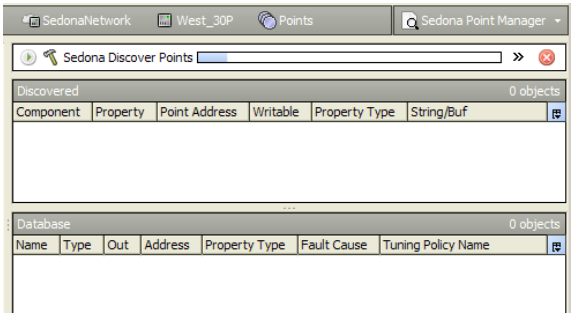


The default discovery type is **Online** when online with the device. Click **OK** to initiate the discover.

Note: **Offline** *is the other discovery type. It is more appropriate for* offline *engineering, and it requires the Sedona device component to have an* associated *app file. For related details, see "Sedona device Association" on page 3-20.*

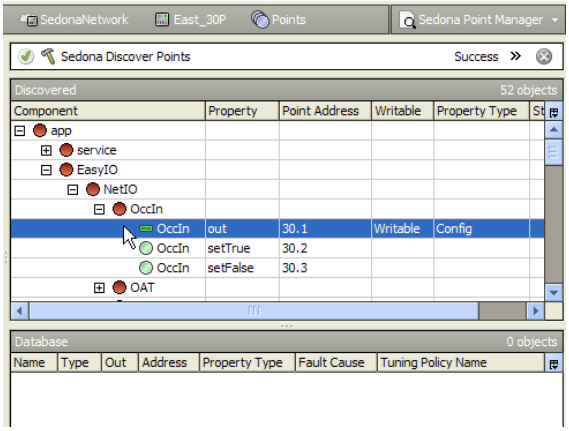
Step 4 A Discover launches a "Sedona Discover Points" job, with progress bar near the top of the view.

Figure 2-9 *Sedona Discover Points job in progress*



The job ends with a single **app** component in the Discovered pane, which you click to expand.

Step 5 Click to select the data items you wish to model in NiagaraAX. Folders and components appear as red orbs, which you expand to see selectable slots—either properties (, ,) or actions ( ).



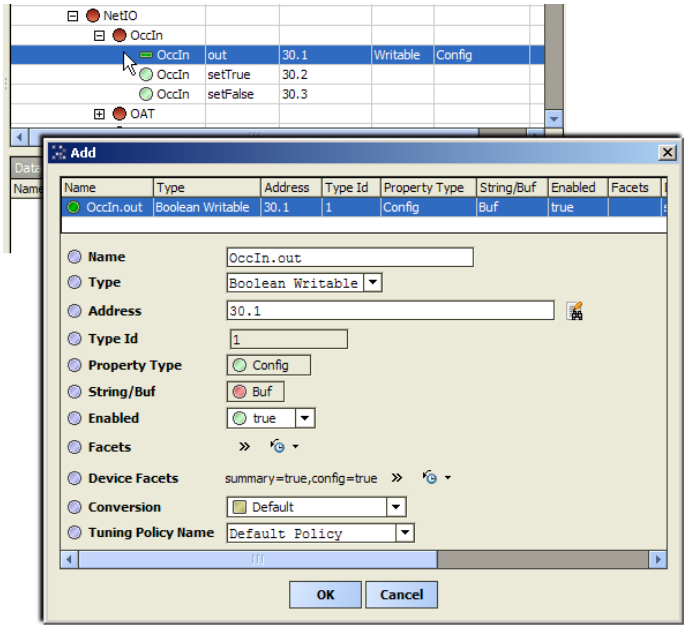
You can map selected items in the station in a number of ways:

- 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 drivers' Point Manager views (however, note if adding multiple items using the last method, all must *either* be properties *or* actions—but *not a combination of both*.)

Step 6 When the **Add** dialog for a *property* appears, you can edit the configuration of that proxy point's SedonaProxyExt before it is added in the Niagara station. Initial property values are determined by Niagara, based upon the property's data type and whether config or runtime. See Figure 2-10.

Figure 2-10 *Example Add dialog for selected property*



The following entries are in the **Add** dialog (and **Edit** dialog) for a Sedona component property:

- **Name** defaults to *SedonaComponentName.propertyName* for example, `OccIn.out` or `AvgTemp.in` This is the Niagara point name only—change if needed (it does *not* affect Sedona component).
- **Type** is the Niagara control point type to use for the proxy point. For most Sedona config types, the default selection is a "writable" control point; whereas runtime types are typically "read-only" points. Float and double data types are Numeric points; integer and long data types default to Enum points. Boolean data types default to Boolean points; and string types are String points. For related details, see "Sedona property data types to default Sedona proxy point types" on page 3-53.
  Note: *Chose Type* before *making changes to other properties below; otherwise they become reset. Unlike other editable entries in the* **Add** *dialog, you cannot edit Type later in the* **Edit** *dialog.*

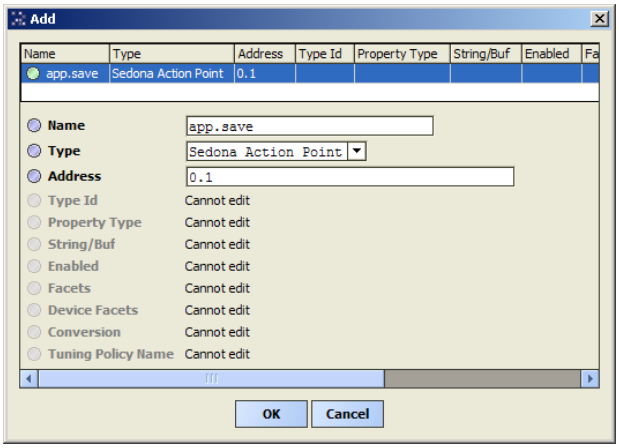
- **Address** is the numerical Sedona address (compID.slotID) of the property slot. It is recommended you do not manually edit this.
- **Type Id** is the numerical type ID of the data.
- **Property Type** reflects whether the value is stored in the device's non-volatile Flash memory (config) or not (runtime).
- **String/Buf** applies only if the target property is type `Sys::Buf`, and determines if Niagara attempts to interpret the byte array as a Sedona string, or just leave it as a byte array.
- **Enabled** is to enable (true, the default) or disable the proxyExt. If set to disabled (false), the point's **out** value has a disabled status.
- **Facets** lets you set the proxy point's facets, where some default facets may exist (e.g., if associated "device facets" are present), or there may be *no facets* defined. Typically, except for string data, you assign facets if they are needed to define units, decimal precision, or states of enumeration, etc. Often, Boolean points use the two facets: `falseText` and `trueText`, and Numeric points use the two facets: `units` and `precision`.
  To assign facets if none exist:
  1. Click the » (open) control for Facets for a popup **Config Facets** dialog.
  2. In the Config Facets dialog, click (time saver) to select from any previously entered facets, and repeat as necessary. If facets needed are not listed, add facets by continuing on to 3.
  3. Click (add) for a new key.
  4. Either type over the highlighted "`key`" name with another key name such as: `units` (if a Numeric point), or `falseText` or `trueText` (if a Boolean point), or click the (drop-down) control for key to select a key name from the list.
  5. Type or use controls as needed for the Type and Value fields, for example for `falseText` or `trueText`, type in a text value in the Value field. Or, for `units`, in Value click the (more) control and in the next popup select the (left side) unit class, then (right side) specific unit.
  6. Click OK to add the facet key/value, and if needed repeat again to add more facets.
- **Device Facets** reflects native facets used in the device, if any.
- **Conversion** specifies the conversion to use between the "read value" (in Device Facets) and the parent point's facets, where "Default" is typically used.
- **Tuning Policy Name** specifies which of the available TuningPolicies in the network should be used to read data from and (if applicable) write data to the property. The "Default Policy" is the default selection—other tuning policies are selectable in the drop-down list. For related details, see:
  - "SedonaNetwork tuning policy notes" on page 3-7
  - "SedonaJen6lpNetwork tuning policy notes" on page 3-27

Step 7 When you have Sedona 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. Within a few seconds, the points will display current values.

Note: *If you selected one or more action slots ( ) instead of properties, the Add dialog is completely different, as shown in Figure 2-11.*

Figure 2-11 *Example Add dialog for action slot on a Sedona component*



In this case, there are only three entry fields, as follows:

- **Name** *defaults to* `SedonaComponentName.actionName` *for example,* `app.save` *or* `UnocHSP.set-` *Where the* `SedonaComponentName` *is always 7 characters maximum. This is the Niagara point name*

only—change if needed (does not affect Sedona component).

- **Type** *is always* **Sedona Action Point**.
- **Address** is the numerical Sedona address (compID.slotID) of the action slot. It is recommended you do not manually edit this.

For more details on Sedona points in NiagaraAX, see the following sections

- "About the Sedona Point Manager" on page 3-47
- "About Sedona proxy points" on page 3-52
- "About Sedona action points" on page 3-55

Configure to allow Sox tunneling

Note: *Starting in Sedona TXS-1.2, the importance of Sox tunneling has diminished in favor of the* **Sox Gateway** *and* **Sedona Tools** *available under each networked Sedona device. However, in some cases Sox tunneling may still be a desired method to open a networked device.*

For details related to the **Sedona Gateway**, *see "Sox Gateway" on page 3-16.*

For details related to **Sedona Tools**, *see "Sedona environment management" on page B-1, with related procedures in "JACE workflow" on page 1-2 and "Installing Sedona environment files in a JACE" on page 1-3 in the "Sedona Framework Network Driver Installation" section.*

You can configure a station with any Sedona Framework network to support "Sox tunneling". This allows (full) Workbench users to open Sox connections to networked Sedona Framework devices, by "tunneling" through the station. However, if the station host is not licensed for Sox tunneling, you can skip this section. See "Workbench and Sedona file requirements" on page 1-1 for related details.

Note: *Complete Sox tunneling details are in the* Sedona Framework Sox Tunneling - Engineering Notes *document. This procedure provides only a few simple steps to get started, and one connection method.*

Add the SoxTunnel service

With the station open in Workbench:

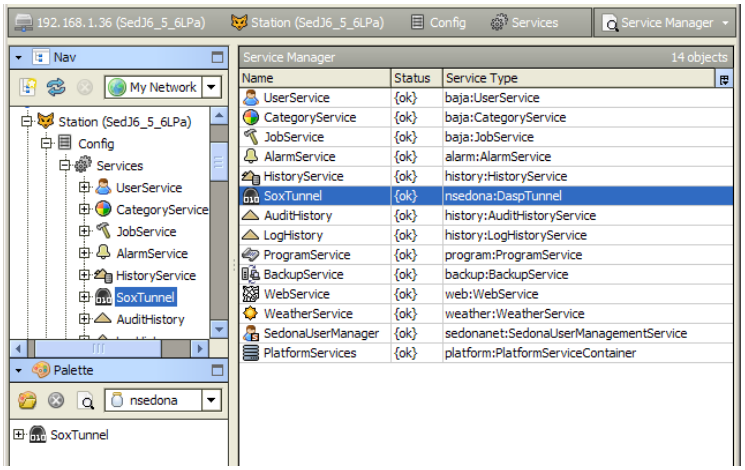
Step 1 In the Nav tree, expand the station's **Config** node to reveal the **Services** container.

If a **SoxTunnel** is already there, skip ahead to Step 4.

Step 2 Open the **nsedona** palette in your Workbench palette side bar.

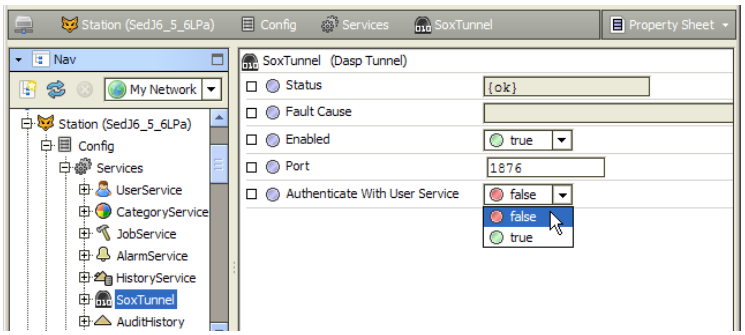
Step 3 From the **nsedona** palette, *drag* (or copy and paste) a **SoxTunnel** into the station's **Services** container. In the popup **Name** dialog, you can rename the service—or, simply use the default name.

The service named "SoxTunnel" (or whatever else), is under your Services container, as shown below.



Step 4 Providing the host is properly licensed for Sox tunneling, the service's status should be "ok".

Typically, only one property may require adjustment: Authenticate with User Service. The default is true.



Set this to false if the password for the Sedona admin user in devices is "blank", or if a non-admin user has different credentials for station login.

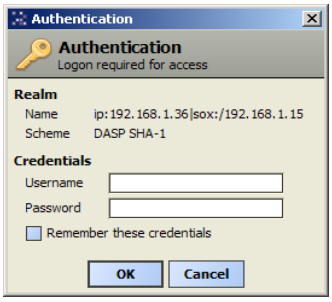
Usually this is the only change required, unless networked Sedona Framework devices are not using the default Sox port of 1876. In that case only, set the **`Port`** property of the SoxTunnel to match, and **`Save`**.

Step 5 To test the SoxTunnel service, in Workbench:

1. Expand the SedonaNetwork or SedonaJen6lpNetwork to reveal child Sedona devices.

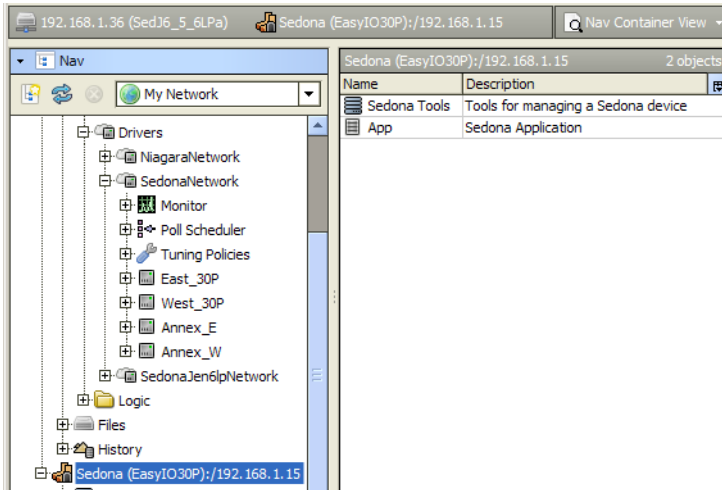
Note: *If selecting a "hibernating" type device, you must first issue a "maintenance mode request" (action) on its SedonaJen6lpDevice, and wait for the popup dialog in Workbench before continuing. However, note that currently Sedona Framework support for hibernating devices is not widely available. For related details see* *"About maintenance mode" on page 3-39.*

2. Right-click a Sedona device, and select **`Open Sox - Tunnel Session`**. An Authentication dialog appears, as shown below.



Step 6 Enter the Username and password of a User in the Sedona Framework app's UserService (for example: the frozen `admin` user), and click **`OK`**. A Sox connection is made tunneling through the running station.

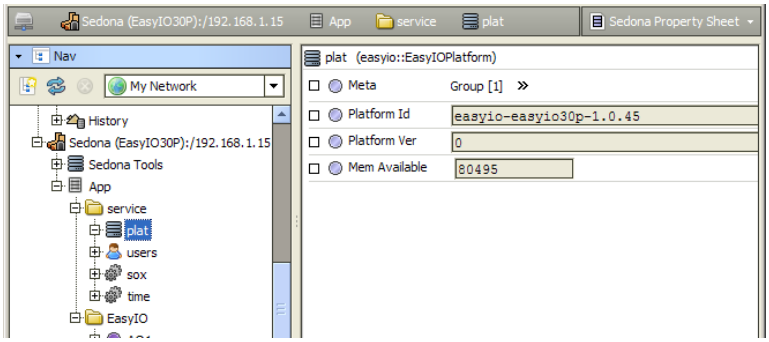
The view space shows the Sedona device's **`Nav Container View`**, with **`Tools`** and **`App`** nodes, as shown below.



Note: *If the connection fails, it is likely because your Workbench host does not have the necessary manifest files for all the kits installed in the Sedona Framework device. In the error, examine any "Details" link to see if missing manifests are named. For related details, refer to* Sedona Manifest Manager - Engineering Notes.

Another possible issue might be a user mismatch between the app's User service and your station login. As shown previously in Step 4, in the JACE station's SoxTunnel service (Config > Services > SoxTunnel), set the "Authenticate with User Service" property to `false` *and* **`Save`**.

As shown above, the tunneled Sox connection also appears in the Nav tree root of that host (on parity with the station). You can also work from the Nav tree, expanding and right-clicking, etc. as needed.



The figure above shows the service folder expanded, with the **`Sedona Property Sheet`** of the "`plat`" component opened in the Workbench view. Click on folders in the App for wire sheet views.

Note: *The* **`App`** *icon changes to add a star whenever you have unsaved changes—meaning they have not been saved to the device's non-volatile Flash memory. Right-click the* **`App`** *and select* **`Actions > Save`**. *Otherwise, changes will remain in device RAM only, and be lost upon a reboot or power event.*

You can also use the **`Sedona Tools`** when Sox tunnel connected to the device. For related details, see the document *Sedona Framework Sedona Tools Guide*.

Chapter 3

Niagara Sedona Framework Concepts

This section describes the NiagaraAX integration of Sedona Framework devices, with these subsections:

Different levels of Sedona data access

Sedona Framework networks in a NiagaraAX station are unique in that (typically) Workbench provides at least two different levels of data access for an integrated Sedona Framework device:

- **The (upper) Niagara component level**
  This level of Sedona data access uses NiagaraAX drivers with components that model the devices, and whatever data items were selected using Sedona proxy points and action points—standard NiagaraAX "station" data. The Niagara station communicates to Sedona Framework devices using the Sox protocol at the application layer, with lower layers either Ethernet/IP (or 802.11 WiFi) if a **`SedonaNetwork`**) or Jennic 802.15.4 / 6LoWPAN (wireless) if a **`SedonaJen6lpNetwork`**.
  This document is mainly about this data access level.
  Note: *To fully support the new Sedona TXS-1.2 features for Sedona device access, it is recommended you install all appropriate "Sedona environment files" used by these devices on the host (e.g. JACE) platform. These include the kit files and platform files, as well as the kit manifest files. Refer to "JACE workflow" on page 1-2 and "Installing Sedona environment files in a JACE" on page 1-3 for details.*
  Otherwise, to be able to discover and add Sedona proxy points, the JACE or Supervisor (station host) requires, at a minimum, the kit manifest file for each Sedona kit that is installed in any networked Sedona Framework device. You can do this in a station connection to the host, using the "Manifest Manager" view on the Sedona Framework network.

- **The (lower) native Sedona component level**
  This level of Sedona data access uses either the "Sox Gateway" and "Sedona Tools" provided under each networked Sedona device, or else a direct Sox connection or a "tunneled" Sox connection from Workbench through the running station.
  Once Workbench is connected to a Sedona Framework device, Sedona provisioning of it may be possible, as well as making Sedona "app" changes to its components, including properties and links.
  Note: *To be able to open a Sedona Framework device in a direct Sox or tunneled Sox connection, the Workbench PC (host) requires the kit manifest file for each Sedona Framework kit installed in the device, as the Sox client is Workbench-based. However, starting in Sedona TXS-1.2 "Sox Gateway" access does not require the Workbench host to have manifest files, as the Sox client is based in the running station. Also a station does not require the SoxTunnel service to support Sox Gateway access. For these reasons, it is expected Sox Gateway access to be used most frequently for Sedona app changes, and Sedona Tools (under each networked device) to be most frequently used for provisioning.*
  For related details, see "Sedona device "tools" views" on page 3-15, "Sox Gateway" on page 3-16. and *"Sedona environment management"* on page B-1.
  In addition, devices based on Sedona Framework 1.2 may have the ability to "host and serve" kit manifests to a Workbench or station client, such that app access can automatically resolve manifest dependencies if not already present in the client's Sedona environment.

- **CHoPAN component access level**
  For SedonaJen6lpNetworks (only), an available "CHoPAN" (or simply Chopan) component access level uses a client-server architecture. Chopan provides Sedona proxy point polling advantages, providing that Jennic-based devices are configured and enabled for Chopan *server* operation.
  Chopan also allows a device's Sedona Framework app to directly access data in other networked devices, or in the JACE station (providing they are Chopan servers) via *client-side* "Chopan points".
  For *hibernating* Jennic-based devices, Chopan client configuration (Chopan points) must be used to access all data, instead of using Sedona proxy points.
  Note: *At the time of this document, Sedona Framework support for hibernating devices (typically battery powered devices) is not widely available. However, the SedonaJen6lpNetwork driver in the NiagaraAX station is "ready" for such device support if this changes.*
  Client-side components in a Jennic-based device's app are added via Niagara access to the JACE station, in the "Chopan Virtual" gateway under its corresponding SedonaJen6lpDevice component. See "About the Chopan Virtual gateway" on page 3-40 for related details. This document summarizes various Chopan-related components and views in the explanation of the SedonaJen6lpNetwork, but complete Chopan details are in the *Sedona Framework Chopan Usage* document.

Caution *In general, before engineering the upper (station) level with Niagara Sedona proxy points and action points, it is recommended to develop and complete the (native) Sedona Framework app running in each Sedona Framework device. Otherwise, data address mismatches between Sedona proxy points in the station and the source Sedona Framework components in a device are likely to occur.*

About Sedona Framework network types

There are two types of Sedona Framework networks possible in a NiagaraAX station: **`SedonaNetwork`** (for Ethernet/IP and/or WiFi-capable Sedona Framework devices) and **`SedonaJen6lpNetwork`** (for 802.15.4 wireless Jennic-based devices). Both network types use the standard NiagaraAX network architecture. See "About Network architecture" in the *Drivers Guide* for general information.

Note: *In previous AX-3.5 and AX-3.4 releases, the* **`SedonaNetwork`** *was sourced from a different module (*`nsedona`*), and could also integrate wireless Jennic-based devices, using a "Jennic6LowpanBridgeService" from the* `jennic` *module. This architecture changed in AX-3.6, with a separate* **`SedonaJen6lpNetwork`** *(*`jen6lp` *module) now used for Jennic-based device integration.*

- *SedonaNetworks created with* AX-3.4 *and* AX-3.5 *are NOT COMPATIBLE with Sedona TXS-1.2 and* AX-3.7*! (However, SedonaNetworks created in TXS-1.1 and* AX-3.6 are compatible.)
- *An offline conversion tool for making a station's* `config.bog` *file compatible with the new Sedona architecture is available. Before upgrading any* AX-3.4 or AX-3.4 JACE *with a SedonaNetwork to* AX-3.7*, you should use this tool to create a compatible station* `config.bog` *file, so that you can install it during the upgrade commissioning process. For details, refer to the document* NiagaraAX Station Migration Tool*.*

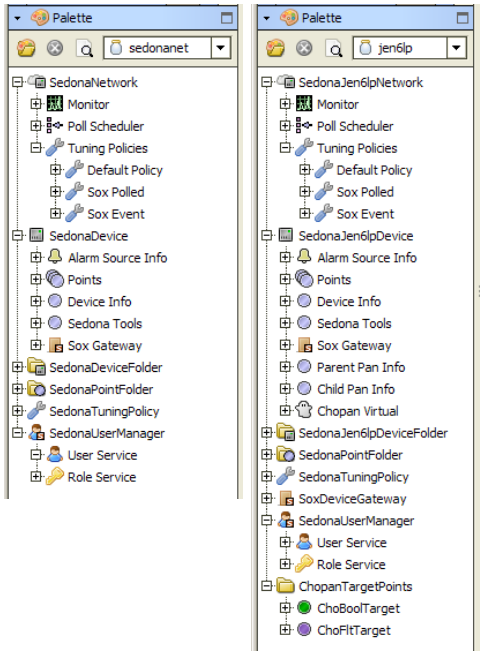
Sedona Framework networks: quick comparison

Sedona Framework network components and child components are found in the

- `sedonanet` palette — for **`SedonaNetwork`**
- `jen6lp` palette - for **`SedonaJen6lpNetwork`**

Table 3-1 compares the palette for each module, as opened in Workbench.

Table 3-1 *Palettes for modules sedonanet (left) and jen6lp (right)*



Note: *As with most other NiagaraAX drivers, you typically do not need to work from a module's palette. Instead, the various Sedona manager views simplify component creation, enforcing proper component hierarchy.*

A possible exception is use of the two "ChopanTargetPoints" in a SedonaJen6lpNetwork, which are "convenience" components that can be useful when making "target points" in the JACE to receive read-only values from a Jennic-based device, via CHoPAN.

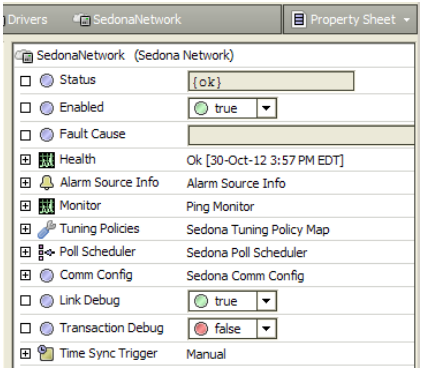
More differences and similarities between the two Sedona Framework network types are provided below:

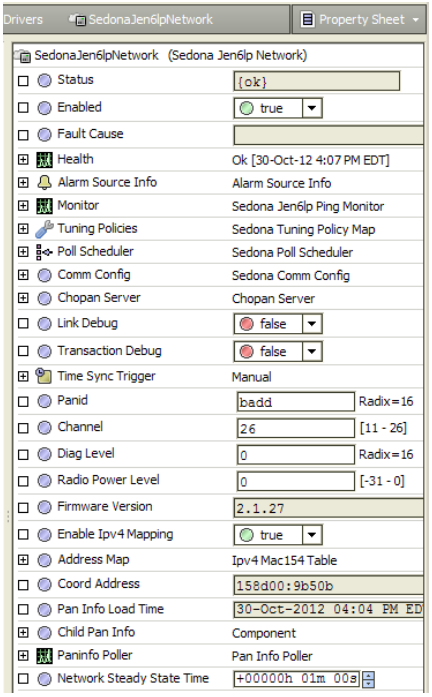
- Sedona Framework network component differences
- Sedona device component differences
- Similarities in Sedona Framework network types

Sedona Framework network component differences

From the *property sheet* of either type of Sedona Framework network, you have access to all the major network-level properties and container slots. See Table 3-2.

Table 3-2 *Property sheet for SedonaNetwork (left) and SedonaJen6lpNetwork (right)*





As shown in Table 3-2, the SedonaJen6lpNetwork (right) has many more properties than the SedonaNetwork (left). Most of these are for the Jennic 802.15.4 configuration of the Sedona Jennic option card in the host JACE controller, for operation as the "Jennic coordinator" for the network.

The SedonaJen6lpNetwork also has "Pan Info" related properties, as do its child SedonaJen6lpDevice components, providing "diagnostic" data useful for troubleshooting wireless Jennic communications.

Currently, the SedonaJen6lpNetwork is also the only Sedona Framework network to use "CHoPAN", reflected by the "ChopanServer" container slot in the network's property sheet. Child SedonaJen6lpDevice components also have a related "Chopan virtual" components. CHoPAN is a light-weight "sessionless" protocol that allows data sharing between devices (and the JACE), without requiring the session-based Sox protocol.

Both Sedona Framework networks have a default "device manager" view for adding and managing child devices. Both networks also have an available "Sedona Device User Manager" view and "Sedona Manifest Manager" view. The SedonaJen6lpNetwork also provides a "Pan Sheet" view for graphically representing the Jennic wireless network's node (tree) hierarchy, including all current "Pan Info" data for the JenNet network.

For more details on properties, actions, and views of Sedona Framework network components, see:

- "About the SedonaNetwork component" on page 3-6
- "About the SedonaJen6lpNetwork component" on page 3-23

Sedona device component differences

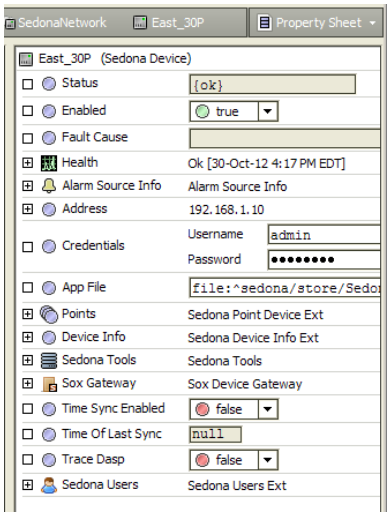
Device types differ between the two Sedona Framework networks—each network has only *one* valid device type:

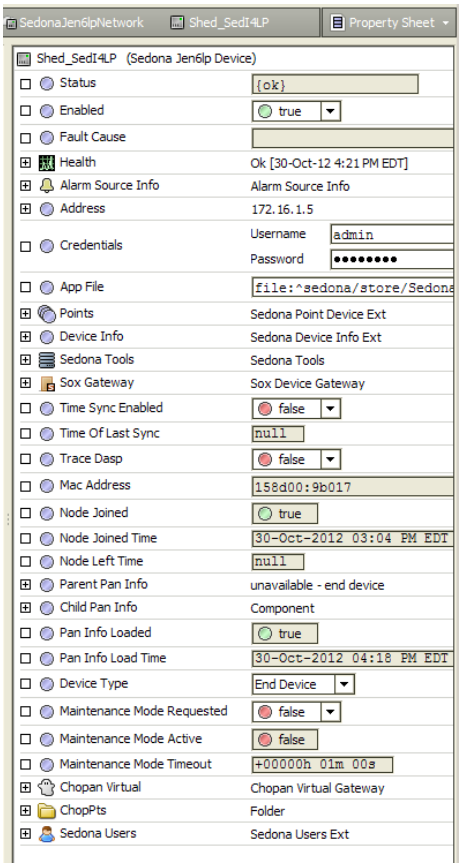
- **`SedonaDevice`** — for SedonaNetwork
- **`SedonaJen6lpDevice`** — for SedonaJen6lpNetwork

You cannot put one device type into the opposite network type.

As shown in Table 3-3, the SedonaJen6lpDevice (right) has more properties than the SedonaDevice (left).

Table 3-3 *Property sheet for SedonaDevice (left) and SedonaJen6lpDevice (right)*





Most additional properties of a SedonaJen6lpNetwork relate to its Jennic 802.15.4 operation, including "Pan Info" and "Maintenance Mode" data. The "Chopan Virtual Gateway" is for modeling the Chopan client interface in the Jennic-based device.

In addition to the standard "Ping" action on the SedonaDevice, a SedonaJen6lpDevice also has "Pan Info" actions and a "Request Maintenance Mode" action.

For more details on properties, actions, and views of Sedona device components, see:

- "About the SedonaDevice component" on page 3-12.
- "About the SedonaJen6lpDevice component" on page 3-36.

Similarities in Sedona Framework network types

Regardless of Sedona network type, any Sedona device component has an important device extension: **`Points`**, for adding/managing Sedona proxy and action points. The **`Sedona Point Manager`** view on any device's Points extension has a a "learn" mode that allows either online or offline point discovery.

Sedona proxy points (and action points) work in the same manner regardless of network type. Although Sedona proxy points in SedonaJen6lp device can use "Chopan" as an alternative to Sox polling or event messaging for updates to the JACE station, the efficiencies provided may mean more with wireless (Jennic) devices that it would with Ethernet/IP capable devices.

Sox gateway access and Sedona Tools access (as well as Sox tunneling) to any networked Sedona Framework device works in the same manner regardless of a device's Sedona Framework network type. This allows "native" Sedona Framework app programming and/or device provisioning. However, if you are accessing a hibernating[1] Jennic-based device in a SedonaJennic6lpNetwork, you must make a "maintenance mode" request (action) on that device to allow Sox access.

1. Sedona Framework support for hibernating devices (typically battery powered devices) is not widely available in the current Sedona Framework TXS release. However, the SedonaJen6lpNetwork driver in the NiagaraAX station is "ready" for such device support, including "maintenance mode" operation.

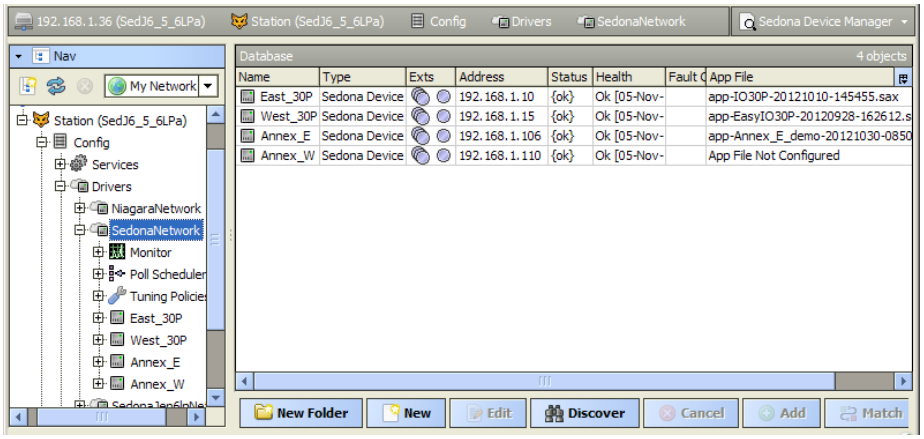
For more details on these topics common among Sedona Framework networks, see:

- "About the Sedona Point Manager" on page 3-47
- "About Sedona proxy points" on page 3-52
- "About Sedona action points" on page 3-55
- "Configure to allow Sox tunneling" on page 2-17

About the SedonaNetwork component

The SedonaNetwork is the top-level component in a station for integration of Ethernet/IP and WiFi capable Sedona Framework devices (only). It is found in the `sedonanet` module. This network is licensed using the "sedonanet" feature, and may have device or point license limits.

Figure 3-1 *SedonaNetwork (default Sedona Device Manager view)*

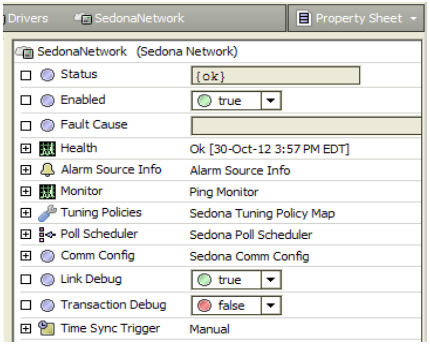


The SedonaNetwork uses the typical NiagaraAX driver architecture, with standard components for device monitor, a poll scheduler, and tuning policies. In addition to the default **`Sedona Device Manager`** view shown in Figure 3-1, there is also an available **`Sedona Device User Manager`** view as well as an available **`Manifest Manager`** view.
The following sections provide more SedonaNetwork details:

- SedonaNetwork properties
  - "SedonaNetwork tuning policy notes" on page 3-7
  - "SedonaNetwork poll scheduler notes" on page 3-7
  - "SedonaNetwork Comm Config" on page 3-7
  - "SedonaNetwork debug properties" on page 3-8
  - "SedonaNetwork TimeSyncTrigger" on page 3-8
- "SedonaNetwork actions" on page 3-8
- "Sedona Device Manager view" on page 3-8
- "Sedona Device User Manager view" on page 3-11
- "Manifest Manager view" on page 3-12

SedonaNetwork properties

Figure 3-2 *SedonaNetwork property sheet*



Among common properties, the SedonaNetwork has the usual collection of Niagara driver status, health, and (ping) monitor properties that work in the standard way. For more details, refer to the *NiagaraAX*

Driver Guide section "Network status properties" and "About Monitor".

Other common properties include the typical container slots for Tuning Policies and a Poll Scheduler. Also the network has two available actions; for details see "SedonaNetwork actions" on page 3-8.

SedonaNetwork tuning policy notes

Proxy points of a SedonaDevice (under a SedonaNetwork) can be assigned to a specific tuning policy, by name. By default, starting in Sedona TXS-1.2, there are *three* differently-named tuning policies is in the network's Tuning Policies container:

- **`Default Policy`** — (Sox Event commType, preselected)
- **`Sox Event`** — (Sox Event commType, fixed)
- **`Sox Polled`** — (Sox Polled commType, fixed)

If needed you can also add additional tuning policies (duplicate, rename, modify properties) and assign to proxy points as needed. For general information about tuning policies and associated properties, see "About Tuning Policies" in the *NiagaraAX Driver Guide*.

By default, these tuning policies of a **`SedonaNetwork`** have the following characteristics:

- The "Write On Start" and "Write On Enabled" properties have been modified to have a default value of *false*. This can help decrease network traffic, which may be critical with some devices.
- Point poll frequency is specified in the tuning policy (similar to how it is done for the BACnet driver).
- A "Comm Type" property specifies how Niagara attempts to update point values, where choices are:
  - Sox Event (default) — uses Sox to subscribe to points, where updates are sent via Sox event messages (similar to COV subscriptions in BACnet). See "Sox Event considerations".
  - Sox Poll — uses Sox read messages to request point values.

Sox Event considerations Although Sox event subscriptions provide efficiencies over normal (Sox Poll) polling from Niagara, be aware that the default "Sox Event" may be inappropriate for some proxy points. Exercise caution if the property has a rapidly changing value, or is a property of a component that has *another property* with a rapidly changing value. Note the other property *does not have to be proxied* to generate events on the proxied property.

Why? A proxy point assigned to a Sox Event tuning policy results in subscription updates triggered by *any value change of any other "same type" property of that component* (type is "runtime" or "config"). This can be a problem if a rapidly-changing property value results in constant event subscription updates, adding unnecessary messaging and possible message fragmentation issues.

For example, consider a proxy point created for the "out" slot of a UI (universal input) component in the Sedona Framework app. Although its value may appear relatively stable, another "raw" property of that component is constantly churning a new A/D (analog to digital) value. Each change results in a new Sox event message update containing all "realtime" property values of that component (the "out" and "raw" values, plus any other realtime property values). This can lead to message bottlenecks on the network, as well as in the Sedona Framework device. In this case, it would be much better to assign a tuning policy using "Sox Poll" for the proxy point for the UI's "out" slot.

Typically, this consideration applies more to "runtime" properties of a Sedona component, and not "config" properties. Ideal proxy point candidates for tuning policies using "Sox Event" comm type are slowly-changing boolean property types, where the parent Sedona component does not have another property with a rapidly changing value.

SedonaNetwork poll scheduler notes

The single Poll Scheduler of a SedonaNetwork has the usual collection of properties. For general information about the poll scheduler, see "About poll components" in the *NiagaraAX Driver Guide*.

SedonaNetwork Comm Config

`Comm Config` in the SedonaNetwork properties specifies communication parameters for Sedona. These parameters affect polls and writes sent to SedonaDevices.

- Retry Count — Number of times a failed message is retried before the transaction is abandoned, where the default value is 3.
- Retry Time — Elapsed time before giving up on a message send attempt (default is 2 sec).
- Session Linger Time — How long unused DASP sessions should be retained before being cleaned up (default=60 sec).

SedonaNetwork debug properties

Among SedonaNetwork properties are two useful for debug purposes. If enabled, and you have station logSetup (in spy) set to "trace" for *both* entries `SedonaNetworkName` and `SedonaNetworkName.sox`, this directs extra messaging to the station's output (visible in platform Application Director):

- Link Debug — For additional trace-level data in station output, useful in tracing the Sox traffic on top of the DASP (Datagram Authenticated Session Protocol) traffic. For example:
  ```
  1:46:29.529 PM|send0:127.0.0.01:r 72 02 00 07 01
  1:46:31:485 PM|rcvd:127.0.0.1:R 52 02 00 07 01 06 42 10 00 00
  ```
- Transaction Debug — For additional trace-level data about Sox transaction management.

SedonaNetwork TimeSyncTrigger

Among SedonaNetwork properties is a "Time Sync Trigger" slot (TimeTrigger component), configurable to send periodic time synchronization messages to networked SedonaDevices. Such devices must support such time synchronization from the station, and be configured as "Time Sync Enabled".

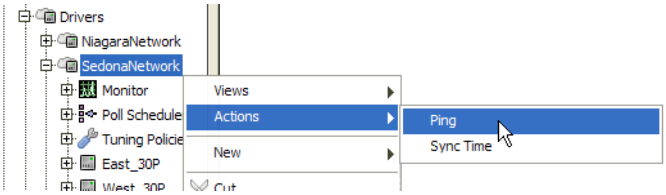
By default, the network's Time Sync Trigger is configured for "Manual" trigger mode. No periodic time sync messages are sent until this is changed to either Daily or Interval (and appropriate ranges are set).

Each SedonaDevice has two related properties and a **`Sync Time`** action. For related details, see "Niagara-to-Sedona device time synchronization" on page 3-18.

Note the network also has a related action; see "SedonaNetwork actions".

SedonaNetwork actions

Figure 3-3 *SedonaNetwork actions*

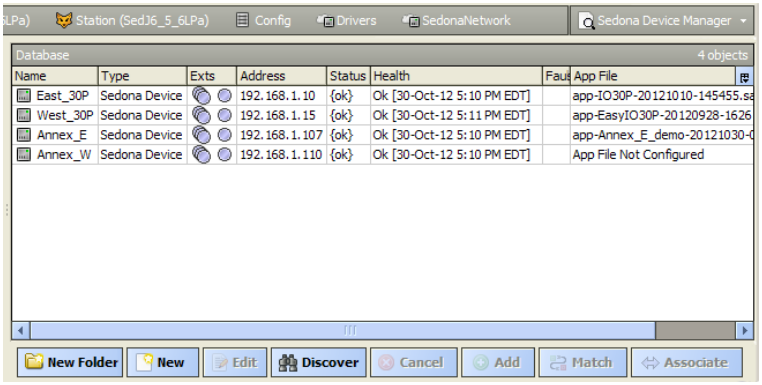


As shown in Figure 3-3, a SedonaNetwork has two available actions, described as follows:

- **Ping**
  (Common among all driver networks) Confirms the network component is seen by the station.
- **Sync Time**
  (New in Sedona TXS-1.2) This invokes the **`Sync Time`** action on each child SedonaDevice component in the network, providing it is configured as "Time Sync Enabled". For related details, see "Niagara-to-Sedona device time synchronization" on page 3-18.

Sedona Device Manager view

Figure 3-4 *Sedona Device Manager view*



The **`Sedona Device Manager`** is the default view on a SedonaNetwork, used to add and edit SedonaDevice components to represent remote Ethernet/IP-capable Sedona Framework devices.

This manager view operates in the standard way as many other NiagaraAX drivers; see "About the Device Manager" in the NiagaraAX Drivers Guide for general information.

Note: *Starting in Sedona TXS-1.2 with AX-3.7, the Sedona Device Manager supports online device discovery for those devices that support it (must implement Sedona 1.2). In addition, support was added for "offline discovery" of proxy point candidates, providing the selected SedonaDevice has an "App File" associated.*

For "quick start" procedures on using the **`Sedona Device Manager`** view to add SedonaDevices, see:

- "Discover and add SedonaDevices" on page 2-2
- "Manually add SedonaDevices" on page 2-5

Other details on the Sedona Device Manager are in the following sections:

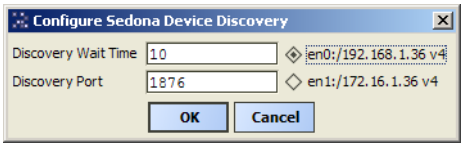
- Sedona Device Manager discovery notes
- Sedona Device Manager data columns

Sedona Device Manager discovery notes

Only Ethernet/WiFi-equipped Sedona Framework devices implemented with Sedona 1.2 may be capable of device discovery in the **`Sedona Device Manager`**. Devices implemented with Sedona 1.0 or 1.1 you must add manually, using the **`New`** button.

To support device discovery, a device's SoxService joins an IPv4 multicast group and listens on port 18760 (the UDP port Niagara always uses to send the request). Devices respond on the "Discovery Port" specified in the configure discovery popup (Figure 3-5).

Figure 3-5 *Configure Sedona Device Discovery popup dialog*



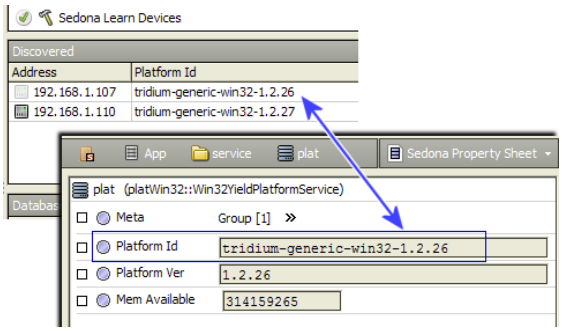
In this dialog:

- **`Discovery Wait Time`** is the number of seconds that Niagara waits to receive Discover responses, at which point the discovery job completes. If no devices are found, the job is marked failed. Depending on the numbers and speed of devices, you may need to increase this time.
- **`Discovery Port`** is typically left at 1876 (default), unless devices' SoxService use a different Port.
- Available Ethernet/IP interfaces are listed on the right, as shown in Figure 3-5 example for a JACE. You can select (click) only *one port* at a time for any discover job.

Note: *On the same* **`SedonaNetwork`***, you can run multiple device discover jobs, selecting a different Ethernet/IP interface if needed. Thus, a single SedonaNetwork may have SedonaDevices installed on different LANs. To help organize, you could use the* **`New Folder`** *button to make* **`Sedona Device Folders`** *first, and then run the Discovers from the* **`Sedona Device Manager`** *of each folder.*

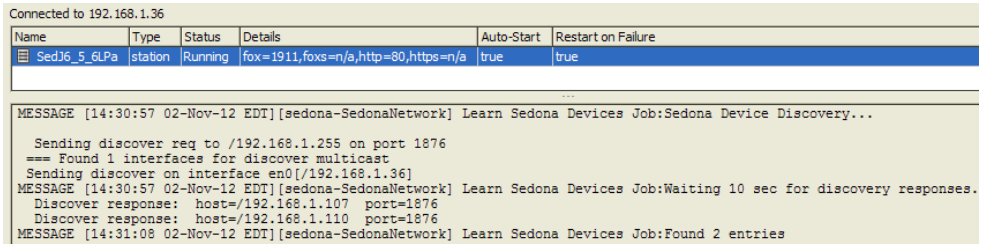
Information from discovered Sedona devices Sedona devices return their **`Platform ID`** value in a discovery response, which appears next to the IP address used by the device in the (top) Discovered pane.

Figure 3-6 *Platform ID value returned with discovered IP address*



As shown in Figure 3-6, Platform Id is a property of the Plat component in the device's App (typically located in a service folder).

Troubleshooting Sedona device discovery Associated with a discover, look in the Application Director of the platform running the station for output similar to below:



As shown in the example successful job above, station output lines indicate the start of the discovery job, the network interface and port used, and finally responses from discovered devices. If no devices respond, make sure the Sedona devices you believe should be discovered are attached on the network.

Note: *If no discovery message is sent by the station, make sure the host platform's TCP/IP settings (subnet mask and default gateway) are set correctly.*

Sedona Device Manager data columns

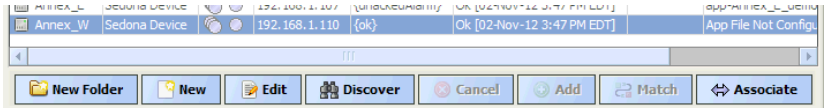
By default, table columns in the Database pane of the Sedona Device Manager include the following:

- **Name**
  Niagara name for the SedonaDevice.
- **Type**
  For the standard SedonaNetwork (`sedonanet` module), always "Sedona Device".
- **Exts**
  Two extensions
  - Points extension for the device — double-click for its **`Sedona Point Manager`** view.
  - Device Info extension for the device — double-click for its property sheet.
- **Address**
  IP address of the Sedona Framework device.
- **Status**
  Device component status, typically "ok" but possibly "fault" (license issue), "down", or "disabled".
- **Health**
  "Ok" or "Fail" with timestamp of last device monitor ping. Fail typically coincides with a down status.
- **Fault Cause**
  Typically blank, unless health is Fail, where any available "Fault Cause" string appears.
- **App File**
  The currently associated Sedona app file in the station's Sedona "app store" file space for this device. Initially, this is "App File Not Configured" for a newly-added (unassociated) device. For related details, see "Sedona device Association" on page 3-20.

Note: *Not included by default, but available in the Table Options (menu), are items "Enabled" and "Credentials".*

Sedona Device Manager buttons

Figure 3-7 *Buttons at bottom of Sedona Device Manager*



Across the bottom of the Sedona Device Manager are the following buttons:

- **New Folder**
  Always enabled. Click to add a new **`SedonaDeviceFolder`**, each with a default **`Sedona Device Manager`** view.
- **New**
  Always enabled. Click to manually add a new **`SedonaDevice`** component, after specifying its address, port, and credentials in a popup **`Add`** dialog. See "Manually add SedonaDevices" on page 2-5.
- **Edit**
  Enabled if one or more SedonaDevice(s) are selected in the Database pane. Click for the **`Edit`** dialog, where all fields (except Type) can be edited.

- **Discover**
  Enabled if the station is online. Click to start a Sedona Device Discover job, where you first choose a discover wait time, target Sox port, and Ethernet/IP port. See "Sedona Device Manager discovery notes" on page 3-9. For a detailed procedure, see "Discover and add SedonaDevices" on page 2-2.
- **Cancel**
  Enabled if a discovery job is in progress; click to cancel.
- **Add**
  Enabled if one or more discovered devices (in top Discovered pane) are selected. Click for an **Add** dialog, where you specify credentials for its app in a popup **Add** dialog.
- **Match**
  Enabled if *one discovered* device (top pane) *and one added device* (bottom pane) *are selected* (as in most Niagara drivers that support online discovery).
- **Associate**
  Enabled if *one* added device in Database pane is selected. Allows you to select a *local* Sedona app file (on your local Workbench PC) to associate with the device in the station's "app store" file space.
  - If online with the station, the file is copied in its Sedona "app store" location for the device. For example: `^sedona/store/SedonaNetwork/Dev1/apps/app-20121030-085058.sax`
    Note: *If also online with the SedonaDevice, after associating an app file to the device, you typically immediately use the* **Application Manager** *(under the device's* **Sedona Tools***) to install (Put) that app in the device.*
  - If programming the station offline, the file is copied in that same spot in the config.bog file, and will be copied to that station upon a station install (Station Copier).

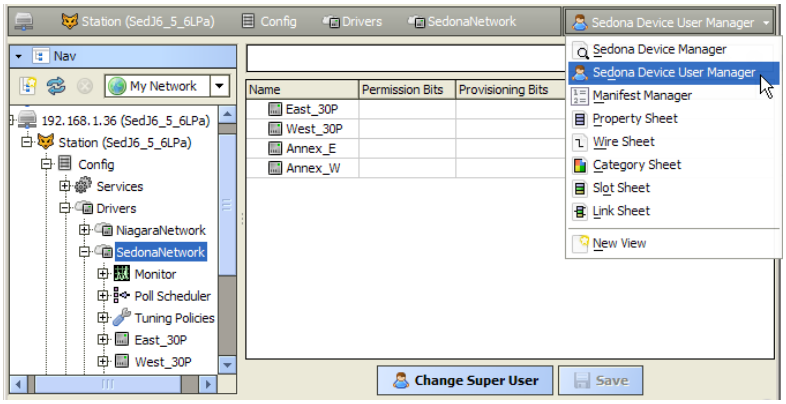
  A device with an associated App File can have proxy points discovered "offline". For related details, see "Sedona device Association" on page 3-20.

Note: *Not included by default, but available in the Table Options (menu), are items "Enabled" and "Credentials".*

Sedona Device User Manager view

Starting in Sedona TXS-1.2, a Sedona network (either a SedonaNetwork or a SedonaJen6lpNetwork) has an available **Sedona Device User Manager** view.

Figure 3-8 *Sedona Device User Manager is an available view on a Sedona network*



Note: *The station requires the* **Sedona User Manager** *(SedonaUserManagementService) copied from the* `sedonanet` *palette into its* **Services** *container to support this function—otherwise, you get an error exception message when accessing this view.*

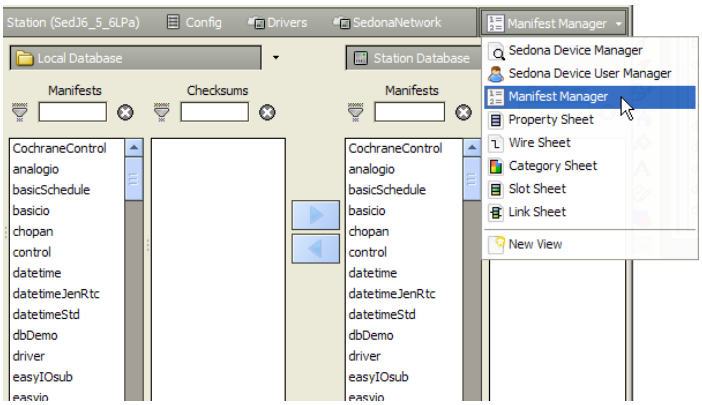
For any usage apart from the global "Change Super User" function, you must first define *Sedona users* and user *roles* using the station's **SedonaUserManager** (service), as default views on **User Service** and **Role Service** child components.

For complete details, see "Sedona users and roles management" on page C-1, including subsections "Sedona user management overview" on page C-2 and "Centrally managing Sedona users" on page C-4.

Manifest Manager view

Note: *Starting in Sedona TXS-1.2, the importance of any Sedona network's* **`Manifest Manager`** *has diminished in favor of using the* **`Sedona Environment Manager`** *platform view. This permits configuration of kit files and platform archive files, in addition to manifest files. However, in some cases the Manifest Manager view on a Sedona network may still be useful (e.g. if a platform connection to the station's host is unavailable). For details on the Sedona Environment Manager, see "Sedona environment management" on page B-1.*

Figure 3-9 *Manifest Manager is an available view on the SedonaNetwork*



As shown in Figure 3-9, the **`Manifest Manager`** is an available view on a SedonaNetwork—note it is the *same view* that is available on a **`SedonaJen6lpNetwork`**.

You can use this view to copy Sedona kit manifest files ("manifests") from your Workbench host (Local Database) to the JACE hosting the station (Station Database). In this case you copy manifests *from the left* side *to the right* side. The station running any Sedona Framework network needs direct access to all kit manifest files for any networked device, to support online Sedona proxy point discovery and proxy point updates. For a related quick start procedure in this document, see "Run the Manifest Manager to ensure kit manifests are loaded" on page 2-12.

However, this view also works "bi-directionally". That is, you can use it to copy manifests *from the* JACE (or Supervisor running the SedonaNetwork) *to your Workbench computer*, copying from right to left. This may be useful if you access the station using a different Workbench host than was used to engineer the station. Note that Workbench requires direct access to the manifests for all kits in a Sedona Framework device when making a Sox connection to it (typically, a Sox tunneling connection)—otherwise, the connection will fail with an exception (exception error "details" usually list the missing manifest file or files).

Note: *Niagara Workbench with Sedona Framework TXS also provides a similar "upper level"* **`Sedona Manifest Manager`** *view, launched from the menu bar:* **`Tools > Sedona Manifest Manager`**. *In that view, the left pane is "Web Database" (source, read-only), while the right (target) pane is your Workbench computer. Providing that you have Internet connectivity, this gives access to manifest files available at* `sedonadev.org`.

Find complete details about using the Manifest Manager in the *Sedona Framework Manifest Manager - Engineering Notes* document.

About the SedonaDevice component

The SedonaDevice is the device-level component representing an Ethernet/IP-capable (or WiFi capable) Sedona Framework device, and can be a direct child of a SedonaNetwork, or else a child of **`SedonaDeviceFolder`** under a SedonaNetwork. This device component contains properties and slots typical to most driver's device components—see "Common device components" in the *Drivers Guide* for general information on device component status properties, health, and alarm source info.

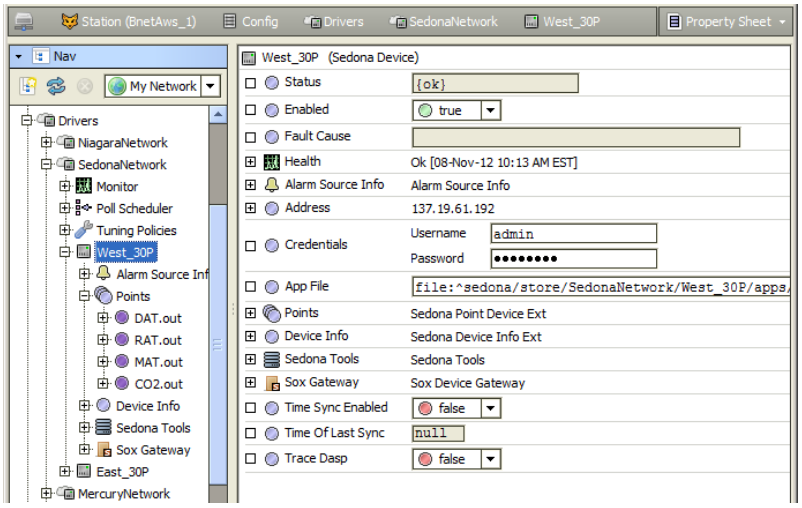
Additional details on other properties, as well as views and other features are in these sections:

- "SedonaDevice properties" on page 3-12
- "SedonaDevice actions" on page 3-14

SedonaDevice properties

Figure 3-10 shows the property sheet view of a SedonaDevice—its default view.

Figure 3-10 *SedonaDevice property sheet*



Properties unique to the SedonaDevice include:

- **Address**
  - Ip — IPv4 address of the Sedona Framework device, reachable by the JACE (NiagaraAX host).
  - Sox Port — The port the device's Sox service is listening on. Default port is 1876.
  - Chopan Port — The port the device's CHoPAN service is listening on. (Currently, CHoPAN is *not applicable* to SedonaDevices under the SedonaNetwork).
- **Credentials**
  The credentials used to authenticate to the device in a Sox connection.
- **App File**
  File ord to an associated Sedona app file (typically `.sax` file) for the device, residing in the file space of the station. Initially, there is no associated app file (device is "unassociated"). However, this field becomes populated in a number of ways. For example, when you Get the app (or Put an app) using **`Sedona Tools`** under the device, the app file is populated. Or, you can "directly" associate an app. For related details, see "Sedona device Association" on page 3-20.
- **`Points`**
  The usual device extension, with a default **`Sedona Point Manager`** view. For related details, see "About the Sedona Point Manager" on page 3-47.
- **`Device Info`**
  The SedonaDeviceInfoExt extension contains read-only properties about a device's Sedona platform type. For more details, see "Sedona Device Info Ext" on page 3-14.
- **`Sedona Tools`**
  For Sedona provisioning tools for this device. See "Sedona device "tools" views" on page 3-15.
  Note: *Before using these tools, installation of "Sedona environment files" are required on the Niagara host (e.g. JACE) running this station, using a Niagara platform connection. This is also required for "Sox Gateway" usage. For details, see "Sedona environment management" on page B-1.*
  For details on the various Sedona Tools, refer to the *Sedona Framework TXS Sedona Tools Guide.*
- **`Sox Gateway`**
  Provides a "gateway" into that device's Sedona app. For details, see "Sox Gateway" on page 3-16.
- **Time Sync Enabled**
  If enabled, allows for periodic (automatic) time synchronization messages to be sent from the Niagara station to this device. The default value is false. For related details, see "Niagara-to-Sedona device time synchronization" on page 3-18.
- **Time of Last Sync**
  Read-only timestamp of when the last successful time sync of the device from the station occurred (whether a periodic message, or from a manual "Sync Time" action). Displays null if no time synchronization has occurred.
- **Trace Dasp**
  If enabled, allows for device-specific tracing of the Sox traffic over the DASP (Datagram Authenticated Session Protocol) traffic.

- **Sedona Users**
  This SedonaUsersExt component registers with the station's SedonaUserManagementService (SedonaUserManager), and *internally* stores the mapping of Sedona users and roles for the device (as defined in the **Sedona User Device Manager** view of the parent Sedona network). This component manages synchronization of those mappings down to the app in the represented Sedona device. For related details, see "Sedona Device User Manager view" on page 3-11.

SedonaDevice actions

Two actions ares available on a SedonaDevice:

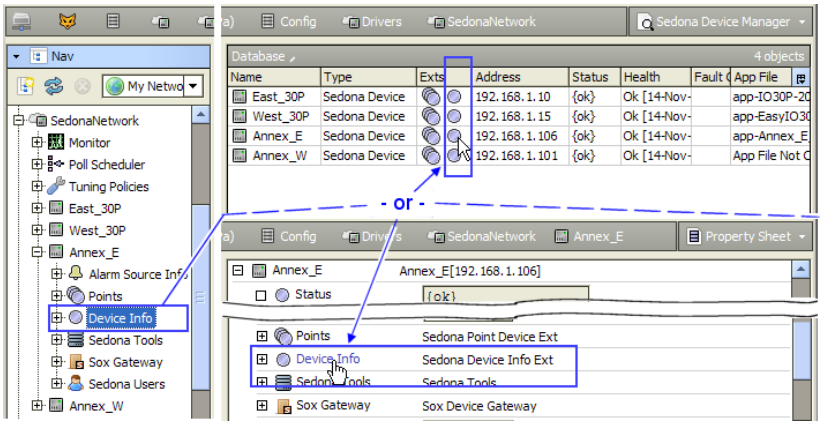
- Ping — related to the typical PingMonitor function, common to most driver networks.
- Sync Time — attempts to synchronize time in the Sedona device with station time. Note this action applies regardless of the value of the SedonaDevice's property **Time Sync Enabled**. For details, see "Niagara-to-Sedona device time synchronization" on page 3-18.

Sedona Device Info Ext

Starting in Sedona TXS-1.2, any Sedona device (SedonaDevice or SedonaJen6lpDevice) has a SedonaDeviceInfoExt (**Device Info** extension). This device extension contains read-only properties about the device's Sedona *platform type*.

As shown in Figure 3-11, you can access this device extension from the Workbench Nav tree, Sedona Device manager, or from the property sheet of the Sedona device.

Figure 3-11 *Sedona Device Info Ext in Nav tree, Device Manager, and Sedona device property sheet*

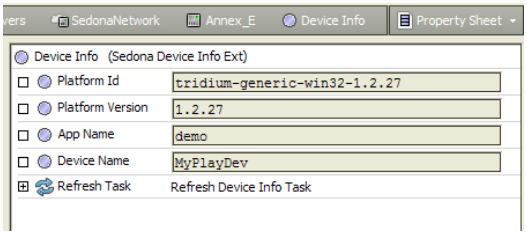


Any method shown above:

- double-click **Device Info** in the Nave tree, or
- click on a Ext in the Device Manager, or
- click on **Device Info** in a property sheet

produces the property sheet for the SedonaDeviceInfoExt. See Figure 3-12.

Figure 3-12 *Property sheet for SedonaDeviceInfoExt*



Properties are read-only status types, read upon station startup (and also whenever a device transitions online from either "down" or "disabled"), and include the following:

- **Platform Id**
  The unique platform ID, a property in the Sedona "**plat**" component in the device's app. The value of this string begins with "*vendor-*", and typically includes platform version information too.
- **Platform Version**
  Another property of the Sedona app's "**plat**" component, reflecting the platform version number.

- **App Name**
  The string value of the App Name property in the device's top-level "**App**" component.
- **Device Name**
  Another property of the Sedona App component (may or may not be implemented in the device).

In addition to the properties above, **Device Info** also contains a child slot:

- **Refresh Task**
  This container has two properties: Status and Fault Cause. It functions to query the Sedona device to refresh the property values (above) in the parent **Device Info** extension, via right-click actions. A refresh may be necessary if a Sedona device has been provisioned (or re-provisioned) since the last station startup. In case a normal attempt to read device info properties fails (upon station startup, or device status transition from "down" or "disabled"), a refresh is *automatically* rescheduled.

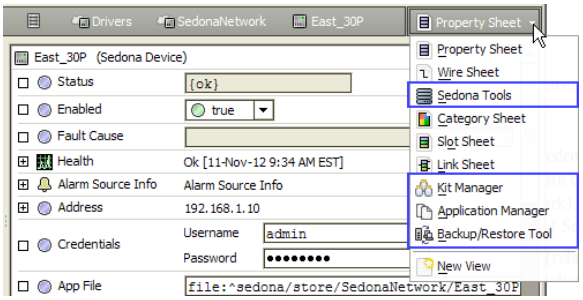
Note: *The* **Sedona Device Info Ext** *has an available* **Refresh** *action, which is equivalent to the* **Execute** *action of its child* **Refresh Task** *component.*

Currently, **Device Info** property values are not available in any manager view (such as the **Sedona Device Manager**), but are available for any *one* selected device using methods shown in Figure 3-11.

Sedona device "tools" views

In addition to the other standard views besides the Property Sheet, Wire Sheet, Category Sheet, Slot Sheet, and Link Sheet, any type of Sedona device (SedonaDevice or SedonaJen6lpDevice) also has additional "Sedona tools" views (Figure 3-13).

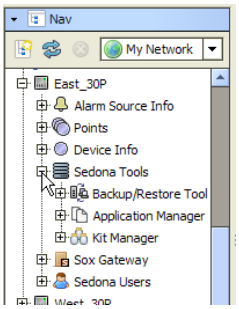
Figure 3-13 *Additional "Sedona tools" views for Sedona device component*



- **Sedona Tools**
  The main "Tools View" for Sedona provisioning of the device "through the station", which lists the installed kits (schema) in the device. Usage requires Sedona environment files installed on the host platform running the station. See "JACE workflow" on page 1-2 and "Installing Sedona environment files in a JACE" on page 1-3, and also "Sedona environment overview" on page B-2 for related details.
- **Kit Manager**
  One of the three Sedona provisioning tools, also available from the **Sedona Tools** view.
- **Application Manager**
  One of the three Sedona provisioning tools, also available from the **Sedona Tools** view.
- **Backup Restore Tool**
  One of the three Sedona provisioning tools, also available from the **Sedona Tools** view.

You can also access these views for a Sedona device in the Nav tree of Workbench (Figure 3-14).

Figure 3-14 *Nav tree access of "Sedona tools" views for Sedona device component*

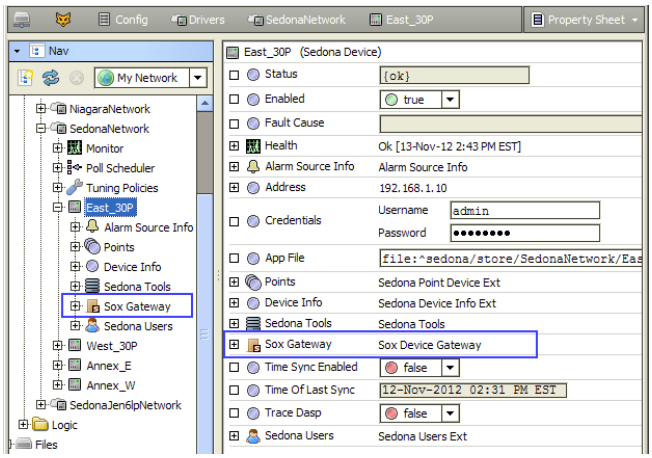


For information on using the Sedona Tools, refer to the *Sedona Framework TXS Sedona Tools Guide.*

Sox Gateway

Starting in Sedona TXS-1.2, each Sedona device component (SedonaDevice or SedonaJen6lpDevice) has a frozen SoxDeviceGateway slot (Sox Gateway).

Figure 3-15 *Sox Gateway is child of every Sedona device*



As shown in Figure 3-15, this gateway appears in the Nav tree under any expanded Sedona device, as well as in the property sheet of the Sedona device.

Prior to the Sedona TXS-1.2 Sox Gateway, to access the Sedona app in a Sedona device you had to either:

- Open a direct Sox connection to the device (**`File > Open > Open Device`**).
  This opened a new Sox session in your Workbench Nav tree root.
- Open a tunneled Sox connection through the station running on the Niagara host (JACE).
  This also opened a new Sox (tunnel) session in your Workbench Nav tree root.

In both cases, the Sox client was running in Workbench. However, now via the Sox Gateway, you can access the Sedona app of a networked Sedona device "seamlessly" in the station (Fox) connection. In this case, the Sox client resides in the Niagara station (JACE).

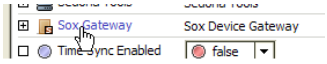
See the following for more details:

- Using the Sox Gateway
- Sox Gateway FAQs

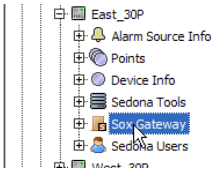
Using the Sox Gateway

To access a device's Sedona app via a Sedona device's Sox Gateway, do one of the following:

- In the property sheet of the Sedona device, simply click the **`Sox Gateway`** slot.

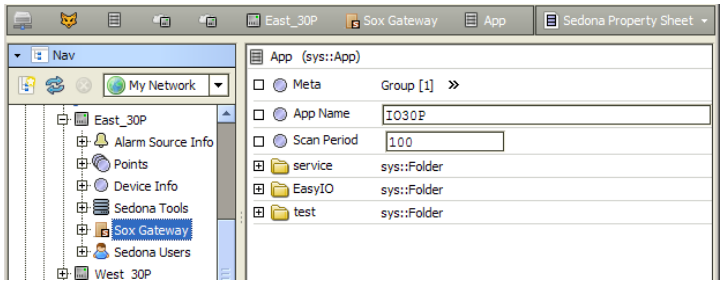


- or, expand a Sedona device in the Nav tree, and *double-click* the **`Sox Gateway`** node.



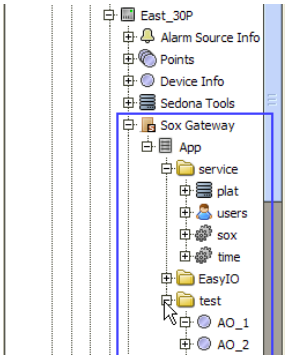
Either method produces the Sedona property sheet view for the app in the device (Figure 3-16), as the entrance to its "Sox gateway space".

Figure 3-16 *Example Sedona App property sheet via the Sox Gateway*



Once this Sox gateway session is active, you can navigate the app by expanding it in the Nav tree, just as in a regular direct (or tunneled) Sox connection to the device (Figure 3-17).

Figure 3-17 *Expanding Sedona App under Sox Gateway of networked Sedona device*



That concludes the "usage portion" of the Sox Gateway, with a few added technical notes as follows:

- The Sox Gateway uses the same credentials from the Credentials property in the parent Sedona device to authenticate the connection. If those credentials are invalid, the gateway connection fails.
- The Sox Gateway keeps the underlying Sox connection *open* until all the following are met:
  1. There are no views/subscribers to any Sedona component in the Sox gateway space.
  2. There are *no expanded nodes in the Nav tree* for Sedona components in the Sox gateway space.

  Once both conditions above are met, after a brief period, the Sox connection automatically closes.
  Note: *Sox connections are considered "expensive" on both the client side (JACE) and server side (Sedona device). Therefore, the recommended practice is for you to fully navigate away from the Sox gateway space when done.*

Sox Gateway FAQs

The following are FAQs (frequently asked questions) on the Sox Gateway:

Q: **Is there special licensing for the Sox Gateway?**

A: Sox Gateway functionality is included for any host (e.g. JACE) licensed to run a Sedona network.

Q: **What is the difference between a Sox Gateway and Sox Tunneling?**

A: There are several differences, but all center around the location of the Sox client that communicates with the remote Sedona device.

Item	Sox Gateway	Sox Tunnel
Sox client	JACE/station	Workbench
Location of Sedona Environment Files	JACE/station	Workbench PC
Requires Sox Tunnel Service in station?	No	Yes
Creates "session nodes" in Workbench Nav tree?	No	Yes

- **Sox Gateway**
  With Sox Gateway access, the Sox client is in the JACE/station. Instead of Workbench connecting to the Sedona device, either directly or tunneled, the *station* itself connects to the Sedona device. In this case, the station does not require (or use) the SoxTunnelService.
  Since you are only making a Fox (station) connection, your Workbench only needs the Sedona-related modules. However, the station's host platform requires the necessary *Sedona environ-*

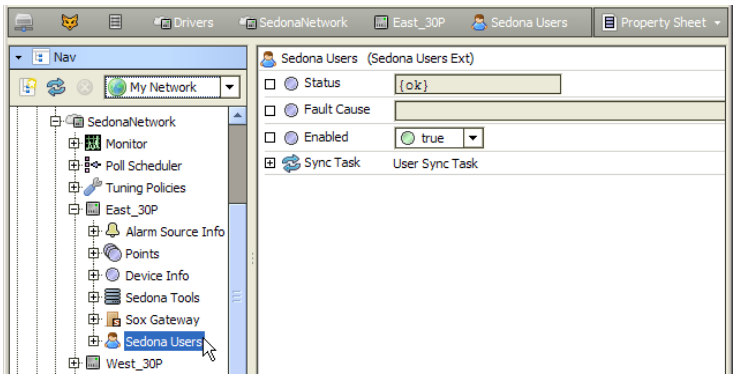
ment files (manifests, kits, platform archives) used by this and other networked devices. The benefit of this approach is that the JACE/station becomes the single point of management for all Sedona devices it has networked to it. See "JACE workflow" on page 1-2 and "Installing Sedona environment files in a JACE" on page 1-3, and also "Sedona environment overview" on page B-2 for related details.
Finally, when accessing the app in a Sedona device through the Sox Gateway, no new "session nodes" are created in Workbench's Nav tree (for example, no Sox session). You can simply navigate to the Sedona device's app. See "Using the Sox Gateway" on page 3-16.

- **Sox Gateway**
  With Sox tunneling, the Sox client is Workbench—that is, Workbench initiates the Sox connection. The "tunnel" capability allows the connection to occur tunneled "through" the JACE, where its station requires the SoxTunnelService. This allows access to Sedona devices not directly reachable.
  However, because the Sox client is Workbench, this requires (at a minimum) your Workbench to have (in addition to Sedona-related modules), all the necessary Sedona *manifests* used by the device in order to make a successful Sox connection.
  Opening a Sox tunnel creates a new "session node" in the Nav tree under the Niagara host/station that you are tunneling through. You manage this session just like any Fox (station) session or "direct" Sox connection, that is: Connect/Disconnect/Close.

Sedona Users device extension

Starting in Sedona TXS-1.2, each Sedona device (SedonaDevice or SedonaJen6lpDevice) has a SedonaUsersExt (**`Sedona Users`** extension).

Figure 3-18 *Sedona Users device extension property sheet*



This extension serves to synchronize changes made in the device's Sedona users, as initiated from other manager views, including the parent network's **`Sedona Device User Manager`** view. No special views are associated with this component (nor are Sedona users represented as child components).

For complete details, see "Sedona users and roles management" on page C-1, and in particular the subsection "User Synchronization" on page C-13.

Niagara-to-Sedona device time synchronization

Starting in Sedona TXS-1.2, each Sedona device component (SedonaDevice or SedonaJen6lpDevice) has two properties related to synchronizing time with Niagara station time:

- **Time Sync Enabled**
  Allows for periodic (automatic) time synchronization messages to be sent from the Niagara station to this device. The default value is false.
  If set to true, automatic (periodic) time synchronization from Niagara can occur. To achieve this, configure the built-in Time Sync Trigger of the SedonaNetwork or SedonaJen6lpNetwork from the default "Manual" (no periodic synchronization) to either "Daily" or "Interval", with the needed values.
- **Time of Last Sync**
  Read-only timestamp of when the last successful time sync of the device from the station occurred (whether a periodic message, or from a manual "Sync Time" action). Displays null if no time synchronization has occurred.

Each Sedona device also has a **Sync Time** action, to send an immediate time synchronization message to the device—note *this does not require Time Sync Enabled to be true*. A "network level" **Sync Time** action is also available—this affects *only* those devices that are configured as Time Sync Enabled.

See the following for details on implementing time synchronization:

- Requirements for time synchronization
- Configuring for time synchronization

Requirements for time synchronization

Time synchronization from Niagara to a Sedona Framework device requires the following:

- The device's Sedona app must include an instance of the `datetime:DateTimeService` component (for example, the **DateTimeServiceStd** from the `datetimeStd` kit, or if a Jennic-based device the **DateTimeServiceJenRtc** from the `datetimeJenRtc` kit). Typically this component is named "time".
- For proper synchronization, you must configure this Sedona **time** component in the device's app with the proper "Utc offset" and "Tz" (time zone) to match the device's time zone. The simplest way to do this is by using that component's default **Date Time Service View**. See "Configuring for time synchronization" for details.

Configuring for time synchronization

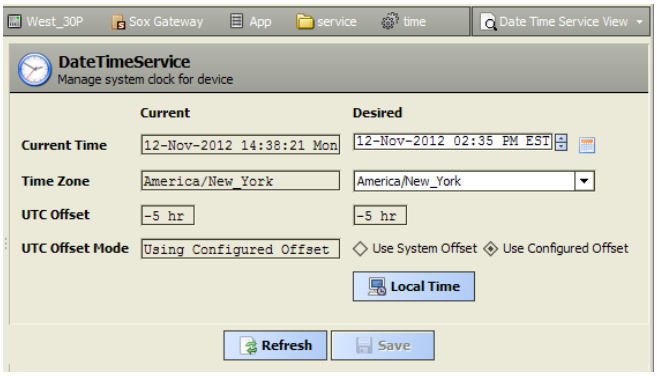
Perform the following procedures for Niagara-to-Sedona time synchronization:

- Configuring the Sedona app for time synchronization
- Configuring Sedona devices in the station for time synchronization

Configuring the Sedona app for time synchronization

Note the app requires a **DateTimeService*Type*** component, for example a **DateTimeServiceStd** or **DateTimeServiceJenRtc** component. Refer to the Sedona device vendor's documentation for related details on which Sedona component and kit is required.

Step 1 Open the device's app and expand to find this "time" service component (typically under "services"), and click on the component to access its **Date Time Service View**.



Step 2 Verify that Current configuration displays "Using Configured Offset" and that the Time Zone matches the time zone configuration of the Niagara station.

If the time zone varies, UTC Offset is incorrect, or "Use System Offset" is configured, do the following under the "Desired" (right side):

1. Select "Use Configured Offset" and then **Save**.
2. Click the drop down control to select Time Zone, then select (or re-select) the *same time zone* as *used by the station*.
3. Click **Save**.
4. Save the App.

Perform other configuration in the Niagara station. See "Configuring Sedona devices in the station for time synchronization".

Configuring Sedona devices in the station for time synchronization

Note: *After configuring a device's Sedona app for time synchronization, you can manually synchronize its time without further configuration in Niagara—simply invoke the right-click* **`Sync Time`** *action on the corresponding Sedona device component (SedonaDevice or SedonaJen6lpDevice) in the Sedona network.*

To configure for *periodic* (automatic) time synchronization from Niagara, do the following:

Step 1 Open the property sheet of the **`SedonaDevice`** or **`SedonaJen6lpDevice`**.

Step 2 Set the **`Time Sync Enabled`** property to `true`, and click **`Save`**.

Step 3 Repeat those two steps for every Sedona device in the network that needs periodic time synchronization.

Step 4 Open the property sheet of the **`SedonaNetwork`** or **`SedonaJen6lpNetwork`**.

Step 5 Expand the **`Time Sync Trigger`** component and change Trigger Mode from "Manual" to either "Daily" or "Interval", and configure the rest accordingly. Click **`Save`** when done.

Step 6 Save the station database.

Automatic time synchronization will now occur at the frequency specified in the Time Sync Trigger component. Note that you can also invoke the **`Sync Time`** action on the network component (SedonaNetwork or SedonaJen6lpNetwork); it results in the **`Sync Time`** action invoked for each networked device that is configured as "Time Sync Enabled".

Sedona device Association

Sedona app file "associations" with Sedona device components is a feature available starting in Sedona TXS-1.2. Among other things, this enables offline point discovery for Sedona devices. The following sections provide more details:

Overview of app file association

Starting in Sedona TXS-1.2, a **`Sedona Environment Manager`** (platform) tool lets you install "Sedona environment files" for Sedona devices networked under a JACE to be located (and maintained) on that JACE platform. Files include kit manifests required for Sox communications, along with kit files and platform archive files needed for Sedona device provisioning. This lets you use the **`Sedona Tools`** under each networked Sedona device. See "Sedona environment management" on page B-1 for details.

Related to this, the JACE also maintains Sedona *app* files in its station's file space for each networked Sedona device. Under this Sedona "app store" file space, each device's "app database" (`apps` subfolder), can hold one or more app files.

Having all these files on the JACE allows Sedona device provisioning to occur directly *from the* JACE station, instead of from Workbench. It also allows tasks like Sedona point discovery to be done from the stored Sedona app file, instead of requiring a Sox connection to the device (as for online point discovery). And finally, it allows a station backup to include Sedona device apps as well.

The act of getting a device's app file onto the JACE, connected with a Sedona device component that represents a specific physical Sedona device is called *association*. See "Association states" and "Association methods" for more details.

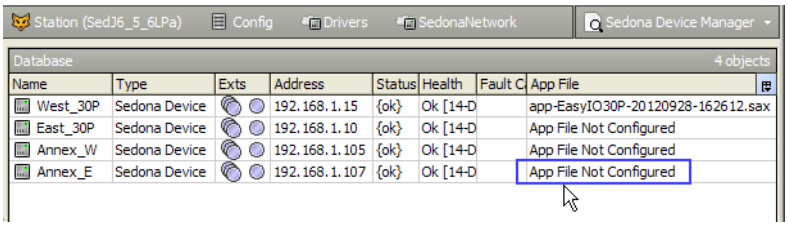
For a procedure on "direct" association, see "Associating the app file for a Sedona device" on page 3-21.

Association states

In a station's Sedona network, each Sedona device component is either "unassociated" or "associated" with a particular Sedona app file. These are the two association *states*.

When you first add a Sedona device component from a network's device manager view, it is typically *unassociated* (unless you copied or duplicated a device component that was associated).

Figure 3-19 *Unassociated device shows "App File Not Configured"*



As shown in Figure 3-19, all but one SedonaDevice shows "App File Not Configured" in the **`App File`** column of the **`Sedona Device Manager`**. This reflects the initial state (unassociated). At this point, you could still perform an *online* point discovery with any of these devices. A Sox connection is made to the device when doing this.

However, you cannot perform an "offline" point discovery with an unassociated Sedona device. An offline point discovery reads the associated Sedona app file to present proxy point candidates (instead of communicating via Sox to the device).

Association methods

A Sedona device component becomes associated with a specific app file in one of two ways: indirectly or directly. See sections "Indirect association" and "Direct association".

Indirect association

App file association automatically results *indirectly* whenever using the **`Application Manager`** view of a device's **`Sedona Tools`**, where the (online) operation is either a:

- Get — The device's running app is *uploaded* and saved as the associated app file (in the device's "app database" in the JACE station's Sedona "app store" file space).
- Put — You select an app in the device's "app database" on the JACE station's Sedona "app store" file space, where it is *downloaded* to the device, overwriting any app currently in the device.

At the end of either operation above, the associated app file matches *exactly* with the app currently in the device. Note this can easily differ in the case of direct association.

Direct association

You can also *directly* associate a Sedona app file on your Workbench PC with any *one* selected networked Sedona device, using the **`Associate`** button in the device manager view for that Sedona network. You can choose either a `.sax` or `.sab` app file to associate.

You can directly associate an app file even if the Sedona device is offline, or even if working in an offline station (`config.bog`) file. Direct association may be most useful for unconfigured devices.

Direct association copies the selected app file to the station's Sedona "app store" file space—under the "app database" for that selected device. In the station's file space, a file ord similar to below is used:

`file:^sedona/store/`*`NameOfNetwork`*`/`*`NameOfSedonaDevice`*`/apps/`*`NameOfSedonaAppFile`*

Example: `file:^sedona/store/SedonaNetwork/Annex_E/apps/Annex_E-20121030-085058.sax`

See "Associating the app file for a Sedona device" for an example procedure.

Note: *Direct association does not transfer (Put) the associated app in the device, or otherwise affect whatever app may be in the device. Use the device's* **`Sedona Tools`**, **`Application Manager`** *tool for such tasks.*

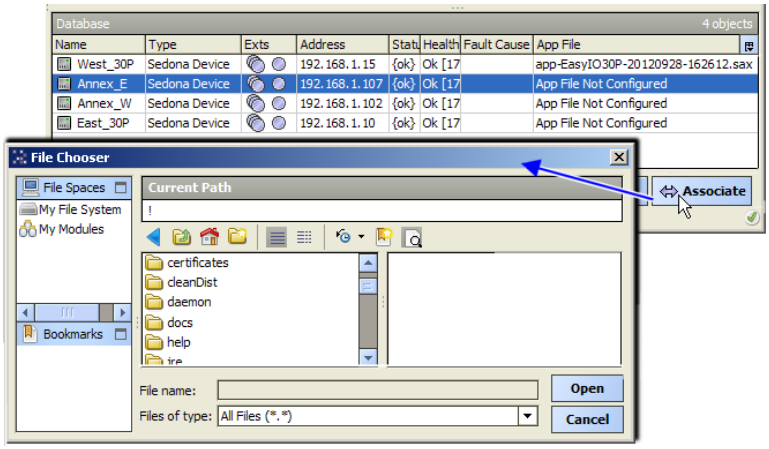
Associating the app file for a Sedona device

The following procedure describes "direct association" of a Sedona app in a device that is currently "unassociated". Note if the device was online and had a configured Sedona app, you could alternatively use the **`Application Manager`** in its **`Sedona Tools`** to Get (save) the app, also associating the device. For related details see "Association methods".

Associating an app file for a Sedona device

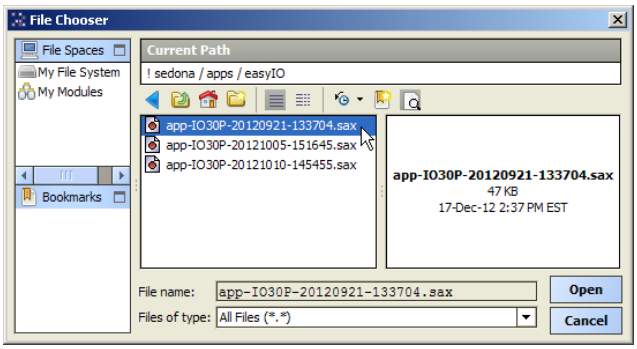
Step 1 In Workbench, open the station and double-click the Sedona network (SedonaNetwork or SedonaJen6lpNetwork) for its device manager.

Step 2 In the device manager, click to select (highlight) a Sedona device, then click the **Associate** button, resulting in a **File Chooser** popup dialog.

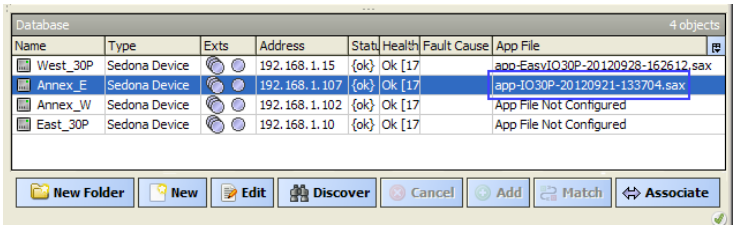


Note: *The* **Associate** *button is unavailable unless you have only a single device selected.*

Step 3 In the **File Chooser**, navigate to the location of the appropriate Sedona app file on your Workbench computer (you can choose either a `.sax` file or `.sab` file to associate), and click **Open**.



This closes the **File Chooser**. This app file is now "associated" with the selected Sedona device component in the network.

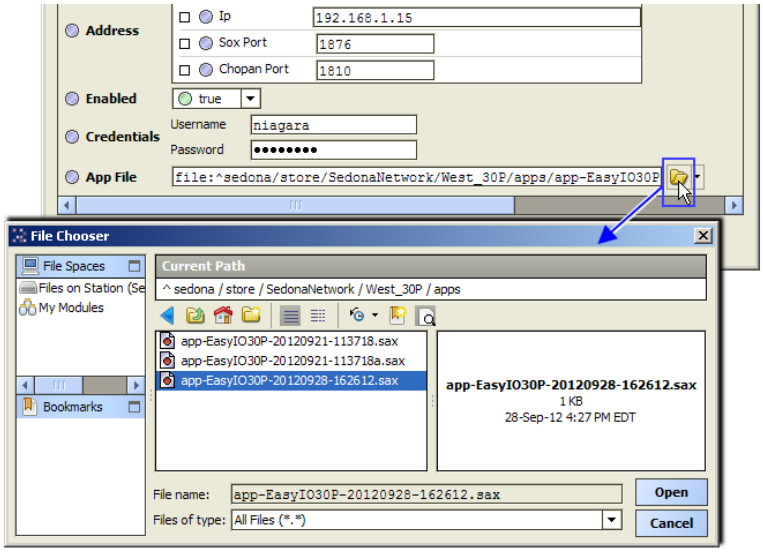


Note the selected app file has been copied into the station's Sedona "app store" file space for this particular networked device. For related details, see "Association methods" on page 3-21.

Selection notes on association

If you have multiple app files stored in the "apps database" for a device in the station's Sedona "app store", you can use the **Edit** device dialog in the device manager to select which one to use. See Figure 3-20.

Figure 3-20 *Using Edit dialog in device manager to pick app file in device's "app database"*



In the **`Edit`** dialog, click the folder icon to the right of the **`App File`** field. By default, the popup **`File Chooser`** initially shows the current associated file highlighted, as shown above.

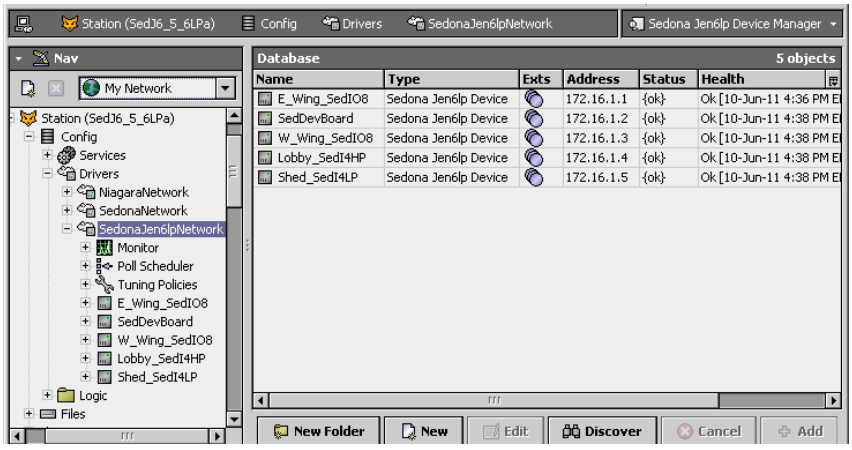
Caution *Remember, changing the associated app file does* not *download (Put) the app in the device, so make sure you are choosing the correct app that represents what is actually in the Sedona device!*

About the SedonaJen6lpNetwork component

The SedonaJen6lpNetwork is the top-level component in a JACE-2, -6, -7 or JACE-x02 XPR station for integration of wireless Jennic-based devices (only), and is found in the `jen6lp` module. This network is licensed using the "jen6lp" feature, and may have device or point license limits.

Note: *The JACE controller requires an installed Sedona Jennic option card to use this network, acting as the Jennic "coordinator" node to the wireless network of devices. Only one Sedona Jennic option card/ SedonaJen6lpNetwork is supported per JACE.*

Figure 3-21 *Sedona Jen6lpNetwork (default Sedona Jen6lp Device Manager view)*



The SedonaJen6lpNetwork uses the typical NiagaraAX driver architecture, with typical components for device monitor, a poll scheduler, and tuning policies. This network component extends the "base" SedonaNetwork with additional slots and views that support the "JenNet" protocol on the RF wireless 802.15.4 WPAN, as well as the 6LoWPAN/IP layer above that.

In addition to the default **`Sedona Jen6lp Device Manager`** view shown in Figure 3-21 and a **`Sedona Device User Manager`** view, there is also an available **`Manifest Manager`** view and **`Pan Sheet`** view on the network. The following main sections provide more SedonaJen6lpNetwork details:

Starting in Sedona TXS-1.2, an additional view is available on the SedonaJen6lpNetwork: the **`Sedona Device User Manager`**. For a summary, see "Sedona Device User Manager view" on page 3-11.

About Jennic-based devices

The term "Jennic-based" device is used in NiagaraAX and Sedona Framework documentation to mean a device based on a Jennic micro-controller, with built-in 802.15.4 wireless connectivity, 6LoWPAN stack support, and the Sedona Framework Virtual Machine (VM). Typical devices are low density I/O modules, thermostats, or other application-specific devices, often installed in places where running communications wiring (back to the JACE) is deemed cost prohibitive.

From a Niagara integration viewpoint, there are two important categories of Jennic-based devices:

- Continuously powered devices — Devices that are always powered, typically from some AC source. Unless noted otherwise, Jennic-based devices are assumed to be in this category.
- Hibernating devices — Devices typically powered by a self-contained battery or batteries. These devices introduce some integration complexity. See "Hibernating devices".

Additionally, it can be helpful to understand details on the underlying Jennic RF network. See "About the Jennic RF network" on page 3-25.

Hibernating devices

Note: *At the time of this document, Sedona Framework support for hibernating devices is* not widely available. *Therefore this section does not apply to most installations. Other sections of this document that mention hibernating devices also note this. However, the SedonaJen6lpNetwork driver in the NiagaraAX station is "ready" for such devices, in case this changes.*

Some Jennic-based devices may be battery-powered, where the only power source is a self-contained battery (or batteries). To conserve battery life, such a device "hibernates" (sleeps) most of the time. At some periodic, configurable interval, the device "wakes up" and executes its routines, which include sending and receiving updates (as needed) on the wireless network. The device then immediately returns to hibernation, consuming almost no power.

While hibernating, the device is unreachable on the Jennic network. Here, the term "wireless" applies to both station communications and power wiring—as there is no power wiring to an external power source. Low power operation is *essential* to minimize the eventual need to replace these batteries.

To support hibernating device integration in Niagara, a special "sessionless" CHoPAN protocol (Compressed HTTP over Personal Area Network) is used for communications between the device and the JACE acting as the coordinator node. Associated are "Chopan client" components, configured and used in the device's Sedona Framework app. In the station, each SedonaJen6lpDevice component includes a special "Chopan Virtual" gateway for building the necessary "Chopan points".

More Chopan details are in other sections of this document, with complete details contained in the document *Sedona Framework Chopan Usage*.

Chopan is used in a Jennic-based hibernating device in solving these challenges:

- Point updates (read or write from/to station)
- Allowing a Sox connection (from Workbench)

Point updates (read or write from/to station) In normal operation, a hibernating device is in hibernation most of the time, and thus a poor candidate for standard Sedona proxy points. *Sedona proxy points should not be used for a hibernating device.* Even if the device is configured as a Chopan server (which is not recommended), and proxy points are assigned to a tuning policy using Chopan "comm type", most poll requests will go unanswered because the device is hibernating. This results in unanswered poll requests, retries, and polling timeouts. Thus, it can adversely affect point polling of other continuously powered devices.

Instead of proxy points, you use the "Chopan Virtual" gateway of a SedonaJen6lpDevice to model "Chopan points" in the device's Sedona Framework app. Chopan points can be either Boolean or Numeric types, and read-only or writable. Additional Sedona Framework app programming is then

required to "link in" Chopan points to other components in the app. In the Niagara station, standard control point "targets" can be made to "receive" writable points from the Sedona Framework app. Conversely, writable Niagara points can write to read-only Chopan points in the hibernating device.

For related details, see "About the Chopan Virtual gateway" on page 3-40.

Allowing a Sox connection (from Workbench) In normal operation, the periodic "awake time" of a hibernating device is often a fraction of a second—again, to conserve precious battery power. However, sometimes a device needs to remain awake, for example when some change is needed in its Sedona Framework app, or if Sedona Framework provisioning (kit changes, backups, etc.) is required.

A Sox connection from Workbench is needed to perform such Sedona Framework operations. When you use Workbench to Sox connect to a hibernating device, the device automatically remains awake for the duration of that session. When you Sox disconnect from the device, it returns back to hibernation.

However, the normal "awake time" of a hibernating device is too short to set up a Sox connection. The device needs a way to "know" that a Sox connection is needed, so it can suspend hibernation for some period, to allow a Sox connection. To provide this, the SedonaJen6lpDevice component in the JACE station (that represents any Jennic-based device) has an available action "Request Maintenance Mode".

For related details, see "About maintenance mode" on page 3-39.

About the Jennic RF network

This section summarizes the underlying architecture used for a network of devices represented in Niagara by a SedonaJen6lpNetwork. Some oversimplifications exist. For more complete details about Jennic 802.15.4 wireless communications, refer to Jennic documentation. At the time of this document, one URL for accessing this information is: www.jennic.com/support/user_guides

Jennic-based device protocol stack

Jennic-based devices use a protocol stack, that from "bottom up" use the following:

- IEEE 802.15.4 at the bottom physical layer and MAC (media access control) layer above—using wireless RF in the 2.4Ghz frequency range. This protocol defines the lower layers of a WPAN (wireless personal area network), including 16 "channels" and the definition of one node being a "full function" type with PAN "coordinator" ability, that is, in charge of the whole network.
- "JenNet", the Jennic protocol that manages network formation and message routing, is the next layer up. A JenNet network has nodes that all use the same "PAN ID" (a two byte hexadecimal number). JenNet uses a "self healing tree" network structure, where all messaging goes up and down the tree. Nodes can have only one parent, but may have one or more children. Peer nodes do not communicate directly with each other, but rather through their parent.
  Every 802.15.4 network has a single coordinator node plus one or more routers and/or end devices. In a Jennic-based network, the JACE is always the coordinator, at the top of tree. Routers are continuously powered Jennic-based devices capable of relaying messages between nodes (typically in addition to performing other application functions). End devices cannot relay messages, and may be Jennic-based hibernating devices.
  Nodes maintain "PAN info" about any children, and routers also maintain this information about their parent. End devices do not store any "PAN info". In the Niagara station, this "PAN info" is available graphically as a view on the SedonaJen6lpNetwork, and numerically in properties of some components. For details in this document, see "Pan Sheet view of SedonaJen6lpNetwork" on page 3-35. For more details, see the engineering notes document *Jennic Network Visualization (Pan Sheet)*.
- Above the JenNet protocol layer is the 6LoWPAN layer, which the Jennic-based device uses to establish an IPv6 address. Through the operation of the JACE coordinator and Jennic-based driver, each child node under the coordinator also automatically has a mapped IPv4 "private network" address. The IPv4 address is used to support Workbench "tunneled Sox" connections to devices.
- Sedona Framework is the application layer, which includes protocols DASP (Datagram Authentication Session Protocol) and the Sox protocol, along with other parts of the Sedona Framework.

Jennic radio operation notes

Because Jennic-based devices operate in the 2.4Ghz band spectrum, common to other wireless devices like 802.11 WiFi, Bluetooth, cordless phones, and baby monitors, a few guidelines may help when configuring a Jennic-based network, to minimize interference. Additional considerations can also help optimize throughput.

Location of devices Jennic-based devices, including the JACE with Sedona Jennic option card, should not be located near noise-generating equipment such as microwave ovens. Other devices based on personal area networks (PAN) may also cause interference, as well as devices previously mentioned.

Use of an antenna extension cable is often required, for a number of possible reasons—often more than one. Examples include to relocate the device's antenna outside of a metal enclosure that houses it, to distance it from the antenna of another device, and/or to optimize reception and transmission.

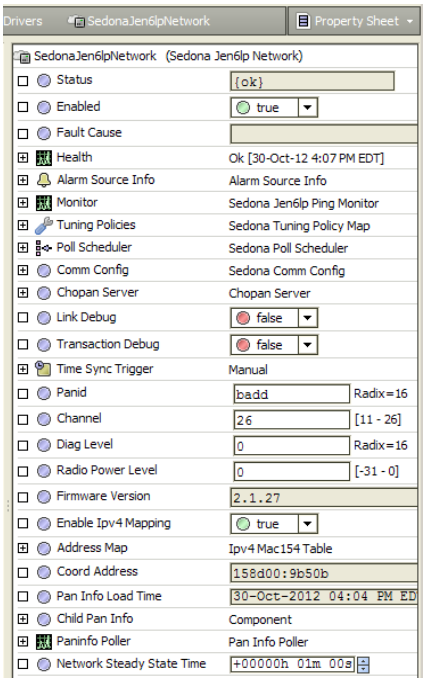
Acceptable distances between the (JACE) coordinator and nodes will vary with a number of factors, including building structure makeup, sources of interference, and so on. Use of the Pan Sheet view on the SedonaJen6lpNetwork can help diagnose the JenNet tree structure and the link qualities between nodes.

SedonaJen6lpNetwork (Sedona Jennic option card) configuration Several network properties can affect radio operation; one example is Channel selection. For more details, see "SedonaJen6lpNetwork coordinator properties" on page 3-29.

SedonaJen6lpNetwork properties

Among common properties, the SedonaJen6lpNetwork has the usual collection of Niagara driver status, health, and (ping) monitor properties. For general details, refer to the *NiagaraAX Driver Guide* section "Network status properties" and "About Monitor".

Figure 3-22 *SedonaJen6lpNetwork property sheet*



Note: *The network's Monitor mechanism, as well as the related "Ping" action on each child SedonaJen6lpDevice, is unique in that* device communications are not used *on a ping. Instead, the check is with the "Joined" status of the device in the JACE coordinator's address map for the JenNet network. See these automatically-added entries by expanding the "Address Map" container slot in the property sheet of the SedonaJen6lpNetwork. For details, see "SedonaJen6lpNetwork coordinator properties" on page 3-29.*

Other common properties include the typical container slots for Tuning Policies and a Poll Scheduler. A single "Ping" action is available on the network.

These sections provide more details on SedonaJen6lpNetwork properties:

SedonaJen6lpNetwork tuning policy notes

Proxy points of a SedonaJen6lpDevice (under a SedonaJen6lpNetwork) are assigned to a specific tuning policy, by name. By default, starting in Sedona TXS-1.2, there are *three* differently-named tuning policies is in the network's Tuning Policies container:

- **`Default Policy`** (Chopan commType, preselected)
- **`Sox Event`** (Sox Event commType, fixed)
- **`Sox Polled`** (Sox Polled, commType, fixed)

If needed, you can add additional tuning policies (duplicate, rename, modify properties) and assign to proxy points as needed. For general information about tuning policies and associated properties, see "About Tuning Policies" in the *NiagaraAX Driver Guide*.

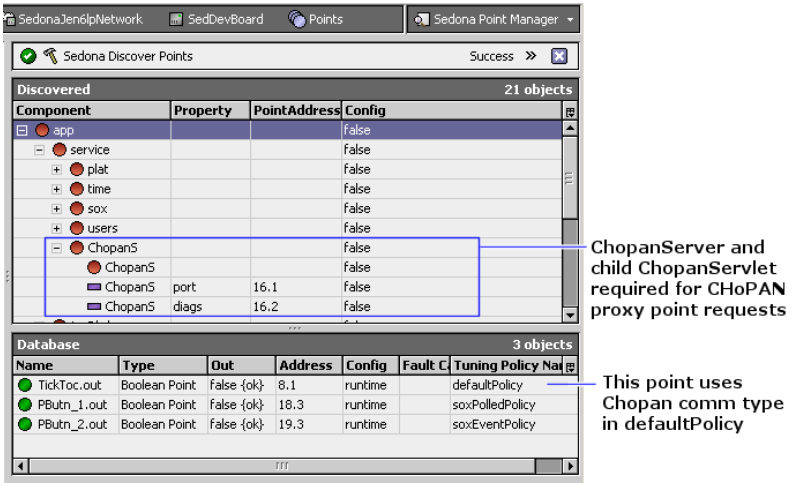
By default, tuning policies of SedonaJen6lpNetwork have the following characteristics:

- The "Write On Start" and "Write On Enabled" properties have been modified to have a default value of *false*. This can help decrease network traffic, which may be critical with some devices.
- Point poll frequency is specified in the tuning policy (similar to how it is done for the BACnet driver).
- A "Comm Type" property specifies how Niagara attempts to update point values, where choices are:
  - Chopan — (default) uses CHoPAN "GET" requests to retrieve point values. This provides efficiencies over other (Sox) comm types, by "batching" multiple reads into a single request.
    Note: *Proxy points using a tuning policy with the Chopan comm type will remain in fault unless the associated Jennic-based device has the chopan kit installed, and its Sedona Framework app is configured to be a Chopan server. For related details, see "Chopan comm type considerations".*
  - Sox Event — uses Sox to subscribe to points, where updates are sent via Sox event messages (similar to COV subscriptions in BACnet). See "Sox Event comm type considerations".
  - Sox Poll — uses individual Sox read messages to request point values.

Chopan comm type considerations Chopan is the most efficient "Comm Type" for a tuning policy, providing a device is configured as a Chopan server. Otherwise, you will need to assign a tuning policy using a comm type of either "Sox Poll" or "Sox Event" to a device's proxy points. Of those two choices, "Sox Poll" is best for rapidly changing values—see "Sox Event comm type considerations" for details.

In the initial point discovery of a Jennic-based device, you can determine if the device supports CHoPAN message requests, by looking in the Sedona Point Manager's "Discovered" pane (Figure 3-23).

Figure 3-23 *Look for ChopanS(ervice) and child ChopanS(ervlet) under App's services node*



Expand the **`app`** node then **`service`** node, and look for "ChopanS" with a child "ChopanS", as shown in Figure 3-23 above. These represent the required ChopanService and ChopanServlet, respectively. If these components are not found, likely the device is not configured to support tuning policies using Chopan.

In this case, proxy points using a tuning policy with "Chopan" as comm type will remain in fault.

Sox Event comm type considerations Although Chopan is typically the most efficient "Comm Type" for a tuning policy, it may be that a target Jennic-based device may not be configured to support it. If not, you must assign proxy points to a tuning policy using another comm types (Sox Event or Sox Poll).

Sox event subscriptions provide efficiencies over normal (Sox Poll) polling from Niagara. However, be aware that the "Sox Event" choice may be inappropriate for some proxy points. Exercise caution if the property has a rapidly changing value, or is a property of a component that has *another property* with a rapidly changing value. Note that the other property *does not have to be proxied* to generate events on the proxied property.

Why? A proxy point assigned to a Sox Event tuning policy results in subscription updates triggered by *any value change of any other "same type" property of that component* (type is "runtime" or "config"). This can be a problem if a rapidly-changing property value results in constant event subscription updates, adding uneccessary messaging and possible message fragmentation issues.

For example, consider a proxy point created for the "out" slot of a UI (universal input) component in the Sedona Framework app. Although its value may appear relatively stable, another "raw" property of that component is constantly churning a new A/D (analog to digital) value. Each change results in a new Sox event message update containing all "realtime" property values of that component (the "out" and "raw" values, plus any other realtime property values). This can lead to message bottlenecks on the network, as well as in the device. In this case, it would be much better to assign a tuning policy using "Sox Poll" for the proxy point for the UI's "out" slot. Even more efficient would be "Chopan" (if supported by the device).

Typically, this Sox Event consideration applies more to "runtime" properties of a Sedona component, and not "config" properties. Ideal proxy point candidates for tuning policies using "Sox Event" comm type are boolean property types, where the parent Sedona component does not have another "like type" property with a rapidly changing value.

SedonaJen6lpNetwork poll scheduler notes

The single Poll Scheduler of a SedonaJen6lpNetwork has the usual collection of properties. For general information about the poll scheduler, see "About poll components" in the *NiagaraAX Driver Guide*.

SedonaJen6lpNetwork Comm Config

`Comm Config` in the SedonaJen6lpNetwork properties specifies communication parameters for Sedona. These parameters affect polls and writes sent to Jennic-based devices.

- **Retry Count**
  Number of times a failed message is retried before the transaction is abandoned, where the default value is 3.
- **Retry Time**
  Elapsed time before giving up on a message send attempt (default is 2 seconds).
- **Session Linger Time**
  How long unused DASP sessions should be retained before being cleaned up (default is 60 seconds).

SedonaJen6lpNetwork Chopan Server

`Chopan Server` in the SedonaJen6lpNetwork properties contains properties to configure the CHoPAN server mechanism for this network in the JACE's station. Networked Jennic-based devices can access station data via "Chopan points" in their apps, as *client requests* to this server. Chopan server operation is required if the network includes any hibernating[1] Jennic-based devices.

Note: *Chopan server operation requires the JACE's* `jen6lp` *license feature to have attribute* `export="true"`.

- **Port**
  UDP port that the JACE station monitors for CHoPAN client requests from networked devices. The default port is 1810.
- **Enabled**
  Enables or disables (the default) the network's Chopan server.
- **Debug On**
  If enabled, *and* you have station logSetup (in spy) set to "trace" for `SedonaJen6lpNetwork.chopanSvr`, extra messaging is directed to the station's output (visible in platform Application Director).
- **Status**
  Status of station's Chopan server, typically "ok" if enabled.
  Note: *Status is fault if the JACE's license feature* `jen6lp` *does not have attribute* `export="true"`.
- **Fault Cause**
  If the Chopan server has a fault status, this text string explains why.

1. Currently, Sedona Framework support for hibernating devices is not widely available. Therefore, use of the JACE Chopan server is not typically required.

SedonaJen6lpNetwork debug properties

Two SedonaJen6lpNetwork properties are useful if troubleshooting. If enabled, *and* you have station logSetup (in spy) set to "trace" for *SedonaJen6lpNetworkName* and *SedonaJen6lpNetworkName*.sox, this directs extra messaging to the station's output (visible in platform Application Director):

- **Link Debug**
  For additional trace-level data about Sox traffic on top of the DASP (Datagram Authenticated Session Protocol) traffic. For example:
  ```
  1:46:29.529 PM|send0:127.0.0.01:r 72 02 00 07 01
  1:46:31:485 PM|rcvd:127.0.0.1:R 52 02 00 07 01 06 42 10 00 00
  ```
- **Transaction Debug**
  For additional trace-level data about Sox transaction management.

SedonaJen6lpNetwork TimeSyncTrigger

Among SedonaJen6lpNetwork properties is a "Time Sync Trigger" slot (TimeTrigger component), configurable to send periodic time synchronization messages to networked SedonaJen6lpDevices. Such devices must support such time synchronization, and be configured as "Time Sync Enabled".

By default, the network's Time Sync Trigger is configured for "Manual" trigger mode. No *periodic* time sync messages are sent until this is changed to either Daily or Interval (and appropriate ranges are set).

Each SedonaJen6lpDevice has two related properties and a **`Sync Time`** action. For related details, see "Niagara-to-Sedona device time synchronization" on page 3-18.

Note the network also has a related action; see "SedonaJen6lpNetwork actions" on page 3-32.

SedonaJen6lpNetwork coordinator properties

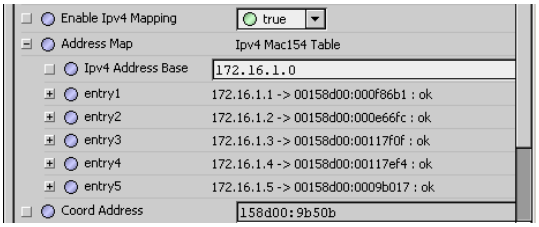
Among SedonaJen6lpNetwork properties are several that apply to the Jennic coordinator operation of the Sedona Jennic option card in the JACE controller:

- **Panid**
  This PAN ID is a two-byte identifier from 0000 to ffff (in hexadecimal) that must be set to match the one in the Sedona Framework app of each Jennic-based device. The default Panid value is "defa". For jobs that have multiple JACEs equipped with a Sedona Jennic option card, it is important to use a different PAN ID for each one (SedonaJen6lpNetwork).
- **Channel**
  The Sedona Jennic option card for a JACE operates in the 2.4Ghz range, on one of 16 selectable channels. Channels are from 11 to 26. Many channels overlap channels used in 802.11 WiFi. To avoid interference from WiFi (in U.S. installations), recommended channels are 15, 20, 25, or 26.
  Note: *Jennic-based devices to be integrated typically have a "Channel Map" property in their Sedona Framework app (e.g. service.plat.rfChanMap) that specifies which of the 16 possible channels it searches (with "beacon" message) for the network coordinator. A best practice is to limit the value of this property to only the channel used by the JACE option card (to which the device will be networked).*
  In jobs with multiple JACEs equipped with a Sedona Jennic option card, it is recommended to use unique channel assignments, if possible, so as not to share bandwidth.
- **Diag Level**
  The level of Jennic/Sedona Framework-related debug messaging to include in the station's standard output (in the platform Application Director view). The value of 0 (default) provides the smallest level. This is read as a bit field, where each bit in the field turns on a different set of debug.
  Note: *The station's log level must have* `sedona.6lp` *and* `sedona.plat.jen6lp` *set to* `trace` *level in order for this property to have any effect. To minimize resource overhead, return log levels back to* `message` *level after debugging.*
- **Radio Power Level**
  This controls attenuation of the RF transmit power level of the coordinator, where the default (0) is highest power. Range is in db, from minimum (-31) to maximum (0).
  In most cases, radio power level is best left at default. If desired, you can set to lower power levels to "simulate" devices that are farther apart or have poor RF connectivity.
- **Firmware Version**
  The firmware version level of the Sedona Jennic option card installed in the JACE.
  Note: *Option card firmware is automatically installed the first time a station starts up with a new or upgraded* `platJen6lp` *module. Typically this adds about 30 seconds to the station startup process.*

Caution *Do not remove power to the JACE during this initial station startup process, for example following an upgrade of software that includes the* `platJen6lp` *module. Otherwise, it could be possible for the option card to lose important data.*

- **Enable Ipv4 Mapping**
  Enables mapping of IPv6 addresses of devices to IPv4 decimal addresses (default is true).
- **Address Map**

Figure 3-24 *Example Address Map entries after enabling network*



Container to specify the IPv4 address network "base" as well as hold entries for discovered nodes:

- **Ipv4 Address Base**
  Private IPv4 subnet for the JACE coordinator to map discovered wireless Jennic-based devices. This subnet must fall within the Class A, B, or C address range, as follows:
  – 10.0.0.0 to 10.255.255.0 (Class A)
  – 172.16.0.0 to 172.31.255.0 (Class B)
  – 192.168.0.0 to 192.168.255.0 (Class C)
  Note: *This must be a different IP subnet than the one used by the JACE. Otherwise, the JACE TCP/IP stack will be unable to route packets appropriately. Also, this must be an unused subnet on the domain. For example, if the domain uses the* `192.168.1.0` *subnet, the default Ipv4 Address Base should not be used. Use a subnet not utilized, say* `10.10.8.0`*, or* `172.16.1.0`*.*
- entry1-***n***
  Automatically added entries for Jennic-based devices discovered by the JACE coordinator. As new wireless devices are discovered, they are assigned the next available IP address. The first address (`n.n.n.0`) is reserved for the coordinator.
  Each of these dynamic entries has the following read-only properties:
  – Ipv4 — Mapped IPv4 address for the device, which is used as its Address when discovered and added as a SedonaJen6lpDevice in the network.
  – Mac154 — The unique 802.15.4 64-bit (8 byte) MAC address of the device, in hexadecimal notation, for example: `00158d00:000f86b1`
  – Joined — Current "JenNet" status with the JACE coordinator, where "true" reflects "ok" device status, else "false" means no communications (device seen as "down").
  Note: *These dynamic entries persist until cleared. Note a "Clear Table" action is available on the Address Map container, which clears all dynamic entries—potentially useful if "starting over" with a new address base.*
- **Coord Address**
  The unique 802.15.4 MAC address of the Sedona Jennic option card (JACE coordinator), in hexadecimal notation without leading zeroes, for example: `158d00:9b50b`

SedonaJen6lpNetwork Pan Info properties

Among SedonaJen6lpNetwork properties are several applicable to JenNet "PAN info" for networked wireless devices, including link quality and statistical data. This data, along with data from child devices operating as router nodes, is used in the network's graphical **`Pan Sheet`** view.

Note: *PAN info requires each router-capable device to have the* `paninfo` *kit. If PAN info is to be read by CHoPAN, the device also requires the* `chopan` *and* `paninfoChopan` *kits. Some minor app configuration is required. Devices that are not router-capable do not require any configuration. Please refer to the "PAN Info Quick Steps" section in the* Jennic Network Visualization (Pan Sheet) - Engineering Notes *document.*

SedonaJen6lpNetwork Pan Info properties include:

- **Pan Info Load Time**
  Timestamp of last successful load of PAN info.

- **Child Pan Info**
  Container component for dynamically-added SedonaJen6lpNeighborEntry (MAC*802.15.4MacAddress*) child components, described below.
  - MAC*802.15.4MacAddress* LQI *n* (for example, `MAC00158d00_00117f0f LQI 174`)
    Contains read-only data about each direct child node under the JACE coordinator, including:
    - Is Sleeping End Device — Whether device is a battery powered (hibernating) end device type (`true`) or a continuously powered device (`false`).
    - Link Quality — Value between 0-255, representing strength of last packet received from this node. Values over 60 represent a good link. Also known as "LQI" (Link Quality Index).
    - Packets Lost — Number of packets sent to device for which an ack(nowledgement) was not received. Note an ack is not expected on all packets.
    - Packets Sent — Number of packets sent to device from coordinator.
    - Packets Received — Number of packets coordinator has received from device.
    - Mac154 — The unique 802.15.4 MAC address of the device, in hexadecimal notation without leading zeroes, for example: `158d00:9b017`
    - Stale — Typically false, else true if this node entry is stale, meaning the device represented is no longer a child. For example, if a device resets and rejoins the JenNet network, it may have a new parent (say a router node). In this case, this Stale values shows true.
- **Pan Info Poller**
  Note: *PAN info polling may be useful for debugging or understanding the network topology, but increases network traffic.*

  Contains configuration and read-only status properties for the "PAN info polling" mechanism that retrieves JenNet PAN info data from child Jennic-based devices on the SedonaJen6lpNetwork.
  - **Enabled**
    Enables PAN info polling if `true`, else if `false` (the default) disables polling.
  - **Use Chopan**
    Whether CHoPAN is used for PAN info polling (the default, true) or if `false`, Sox messaging is used for PAN info polling.
  - **Desired Poll Period**
    Desired PAN info poll period between each device, in hours, minutes, seconds. Default is minutely, that is `00000h 01m 00s`. Note this varies from the "standard" poll scheduler in drivers, where the poll period is the time to complete the entire poll cycle (all devices). See the "Actual Poll Period" property for statistics on the entire poll time.
  - **Statistics Start**
    Read-only timestamp of the start of all accumulated PAN info statistics.
  - **Node Count**
    Read-only count of both the "current" nodes and "average" nodes with polled PAN info.
  - **Average Node Poll Time**
    Time in milliseconds for average polled node.
  - **Inter Node Delay**
    Time in seconds between PAN info polling one device to another device.
  - **Busy Time**
    Percentage of PAN info polling time to total uptime of network.
  - **Actual Poll Period**
    Average time for PAN info polling to complete once for all devices.
  - **Debug**
    If set to `true` enables debug of PAN info polling; the default is `false`.

Note: *Pan Info Poller has three available actions, described as follows:*

- *Enable — set the Enabled property to true.*
- *Disable — set the Enabled property to false.*
- *Reset Statistics — reset Pan Info Poller statistics (as above, including Statistics Start, Average Node Poll Time, Inter Node Delay, and so on).*

SedonaJen6lpNetwork Network Steady State Time property

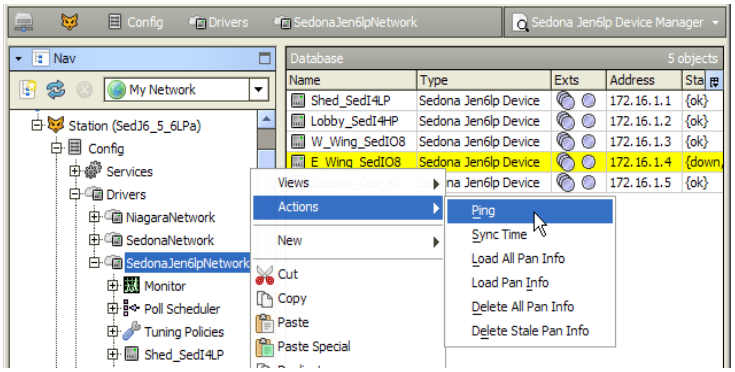
Among SedonaJen6lpNetwork properties is this one, found near the bottom of the property sheet:

- **Network Steady State Time**
  Allows specifying the time after station startup (say from a reboot) before the Sedona Jennic network is considered in a "steady state". Until this timer expires, various service threads do not run, including device ping monitor, proxy point polling, PAN info polling, and automatic writes to proxy points

(such as from tuning policy rules like “Write On Start” and “Write On Up”). This allows all devices to join the 6LoWPAN network before initiating application-level communications.
The default time is 1 minute (`00000h 01m 00s`). In cases of large and/or “deep” networks, adjusting upwards may prevent “write fault” or similar errors from occurring just after station startup.

SedonaJen6lpNetwork actions

Figure 3-25 *SedonaJen6lpNetwork actions*



In addition to the typical “Ping” action, the SedonaJen6lpNetwork has four actions related to “PAN info”. Actions on the network include the following:

- **Ping**
  Verifies station communications with the coordinator (Sedona Jennic option card).
- **Sync Time**
  (New in Sedona TXS-1.2) Attempts to synchronize time in the Sedona device with station time. Note this action applies regardless of the value of the SedonaJen6lpDevice’s property Time Sync Enabled. For details, see “Niagara-to-Sedona device time synchronization” on page 3-18.
- **Load All Pan Info**
  Forces a reload of PAN info from the coordinator and all devices with PAN info (routers).
- **Load Pan Info**
  Forces a reload of PAN info from the coordinator only.
- **Delete All Pan Info**
  Removes all station PAN info from the coordinator and all devices with PAN info (routers).
- **Delete Stale Pan Info**
  Removes all stale station PAN info from the coordinator and all devices with stale PAN info (routers).

For related details, see:

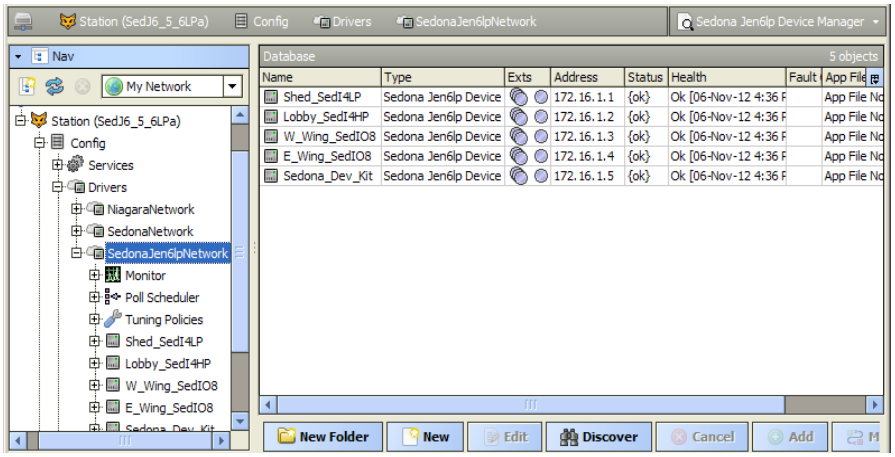
- “SedonaJen6lpNetwork Pan Info properties” on page 3-30
- “Pan Sheet view of SedonaJen6lpNetwork” on page 3-35

Note: *Also refer to the* Jennic Network Visualization (Pan Sheet) - Engineering Notes *document.*

Sedona Jen6lp Device Manager view

The **`Sedona Jen6lp Device Manager`** is the default view on a SedonaJen6lpNetwork, used to discover, add, and edit SedonaJen6lpDevice components that represent wireless Jennic-based devices.

Figure 3-26 *Sedona Jen6lp Device Manager view*



This view operates similar to the device manager in other NiagaraAX drivers that provide online discovery (Learn Mode)—see "About the Device Manager" in the *NiagaraAX Drivers Guide* for general information. Included are available "device folders" using the **`New Folder`** button, where each folder has its own **`Sedona Jen6lp Device Manager`** view. Also provided is the ability to discover and then **`Match`** a discovered device to an existing (and possibly replicated) SedonaJen6lpDevice component.

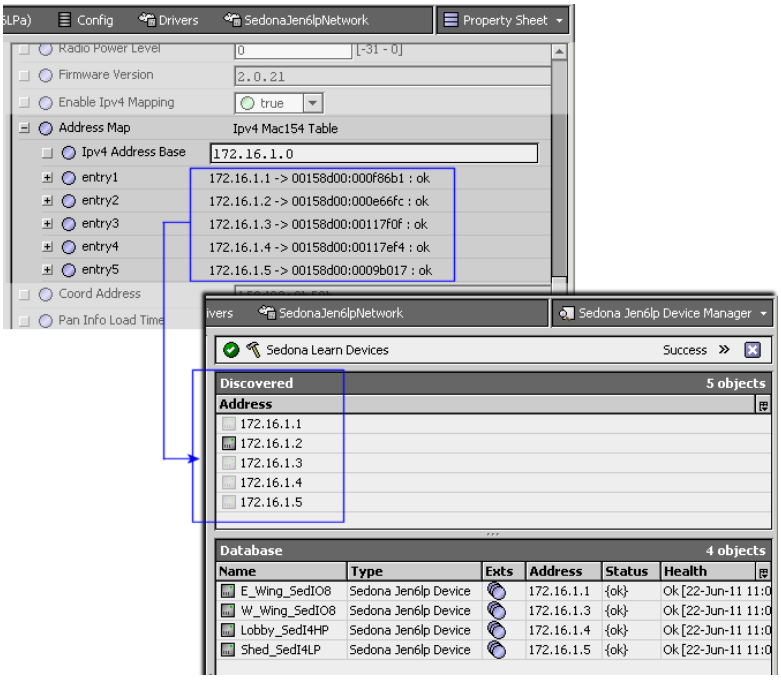
Starting in Sedona TXS-1.2, an **`Associate`** button is also available when any single Jen6lpDevice is selected. For related details, see "Sedona device Association" on page 3-20.

Other details on the Sedona Jen6lp Device Manager are in the following sections:

- Jen6lp Device Manager Learn Mode notes
- Jen6lp Device Manager data columns

Jen6lp Device Manager Learn Mode notes

Figure 3-27 *Address Map entries (of network) determine Discover results in Jen6lp Device Manager*



When you click . **`Discover`** to enter learn mode, and the "Sedona Learn Devices job" runs, it is different than most other drivers, as this does not initiate online messaging. Instead, the learn job reads the "Address Map" entries (mapped IPv4 addresses) of the SedonaJen6lpNetwork component, and for any listed as "ok" (Joined), they appear in the top Discovered pane of the view. See Figure 3-27.

Typically you initially add device components by their default name, for example "dev 172.16.1.2", and then later rename the devices going by your recorded list of MAC addresses that associates each to a particular known device (and installed location).

Note: *For step-by-step details on using this view, see "Discover and add SedonaJen6lpDevices" on page 2-9.*

Jen6lp Device Manager data columns

The Sedona Jen6lp Device Manager has a tabular **`Database`** pane at all times, and while in "Learn Mode", above that a **`Discovered`** pane, as shown in Figure 3-27.

Currently the only data column in the **`Discovered`** pane is:

- **Address**
  Remapped IPv4 address of each Jennic-based device, from the dynamically created entries under the SedonaJen6lpNetwork's "Address Map" container slot.

Table columns in the **`Database`** pane of the Sedona Jen6lp Device Manager (by default) include the following:

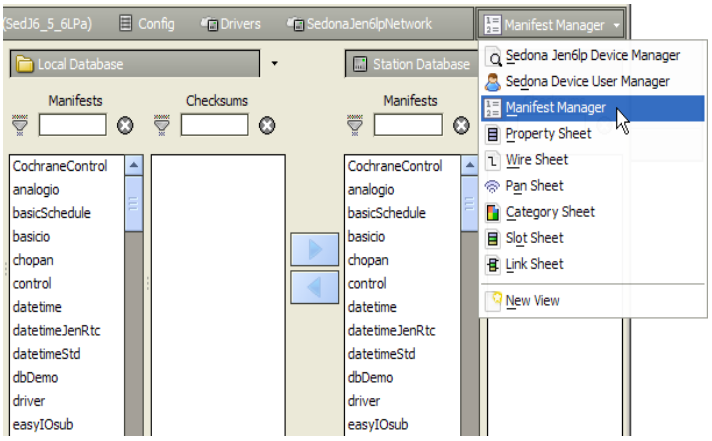
- **Name**
  Niagara name for the device. Default is "`dev IPv4 address`", for example: `dev 198.162.1.4`
- **Type**
  Always "Sedona Jen6lp Device" (cannot modify).
- **Exts**
  Points extension for the device — double-click for its **`Sedona Point Manager`** view.
- **Address**
  Remapped IPv4 address of the Jennic-based device.
- **Status**
  Device component status, typically "ok" but possibly "fault" (license or other issue) or "down".
- **Health**
  "Ok" or "Fail" with timestamp of last device monitor ping. Fail typically coincides with a down status.
- **Fault Cause**
  Typically blank, unless health is Fail, where any available "Fault Cause" string appears.
- **App File**
  The currently associated Sedona app file in the station's Sedona "app store" file space for this device. Initially, this is "App File Not Configured" for a newly-added device. For related details, see "Sedona device Association" on page 3-20.

Note: *Not included by default, but available in the Table Options (menu) of the Database pane, are items "Enabled" and "Credentials".*

Manifest Manager view of SedonaJen6lpNetwork

Note: *Starting in Sedona TXS-1.2, the importance of any Sedona network's* **`Manifest Manager`** *has diminished in favor of using the* **`Sedona Environment Manager`** *platform view. This permits configuration of kit files and platform archive files, in addition to manifest files. However, the Manifest Manager view on a Sedona network may still be useful (e.g. if a platform connection to the station's host is unavailable). For details on the Sedona Environment Manager, see "Sedona environment management" on page B-1.*

Figure 3-28 *Manifest Manager view is one available view on the SedonaJen6lpNetwork*



As shown in Figure 3-9, the **`Manifest Manager`** is an available view on a SedonaJen6lpNetwork—note it is the *same view* that is available on a **`SedonaNetwork`**.

You can use this view to copy Sedona kit manifest files ("manifests") from your Workbench host (Local Database) to the JACE hosting the station (Station Database). In this case, you copy manifests *from the left* side *to the right* side. The station running any Sedona Framework network needs direct access to all kit manifest files for any networked device, to support Sedona proxy point discovery and point updates. For a related procedure, see "Run the Manifest Manager to ensure kit manifests are loaded" on page 2-12.

However, this view also works "bi-directionally". That is, you can use it to copy manifest *from the* JACE running the SedonaJen6lpNetwork *to your Workbench computer*, copying from "right to left". This may be useful if you access the station using a different Workbench host than was used to engineer the station. Note that Workbench requires direct access to the manifests for all kits in a Sedona Framework device when making a Sox connection to it (typically, a Sox tunneling connection)—otherwise, the connection will fail with an exception (exception error "details" usually list the missing manifest file or files).

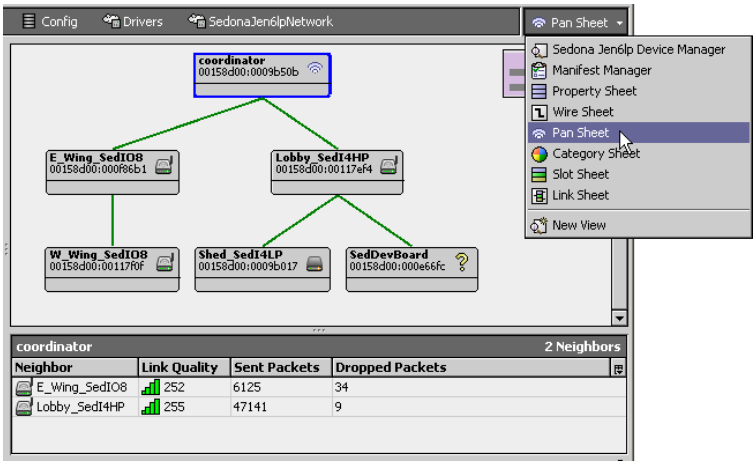
Note: *Niagara Workbench with Sedona Framework TXS also provides a similar "upper level"* **`Sedona Manifest Manager`** *view, launched from the menu bar:* **`Tools > Sedona Manifest Manager`**. *In that view, the left pane is "Web Database" (source, read-only), while the right (target) pane is your Workbench computer. Providing that you have Internet connectivity, this gives access to manifest files available at* `sedonadev.org`.

Find complete details about using the Manifest Manager in the *Sedona Framework Manifest Manager - Engineering Notes* document.

Pan Sheet view of SedonaJen6lpNetwork

As shown in Figure 3-9, the **`Pan Sheet`** is an available view on a SedonaJen6lpNetwork—note this view is unique to the SedonaJen6lpNetwork.

Figure 3-29 *Pan Sheet is one available view on the SedonaJen6lpNetwork*



This view is a diagnostic tool that can *graphically show* the "self healing tree structure" of the underlying JenNet wireless network, along with associated "PAN info" statistics stored in the (JACE) coordinator and any devices operating as "router" nodes. Included in statistics is the radio "Link Quality", representing strength of last packet received. The (JACE) coordinator node is always at the top of the network tree.

Nodes in the view's top pane are selectable, with context-sensitive PAN info data about neighbor nodes (children and/or parent) shown in the table at the view bottom. Data from router nodes is retrieved by the "PanInfo Poller" mechanism of the SedonaJen6lpNetwork (once enabled), or by explicit "Load Pan Info" actions. Related PAN info actions are on the network component, as well as any SedonaJen6lpDevice component.

Note: *Usage requires installing one or more Sedona Framework kits in router-capable devices (only), and some minimal app configuration in those devices. Additionally, in the station each SedonaJen6lpDevice should have its "Device Type" property set (from the default "Unknown") to either "Router" (if a router-capable device) or "End Device" if not capable of router operation.*

Note that PAN info polling is not recommended for continuous operation, as it consumes network bandwidth. However, you can still use the Pan Sheet without polling, using the "Load All Pan Info" action on the SedonaJen6lpNetwork. In general, Pan Sheet usage is helpful for initial network setup and/or troubleshooting communications problems.

Find complete details about using the **`Pan Sheet`** in the *Jennic Network Visualization (Pan Sheet) - Engineering Notes* document.

For other related sections in *this* document, see:

- "SedonaJen6lpNetwork Pan Info properties" on page 3-30
- "SedonaJen6lpNetwork actions" on page 3-32

About the SedonaJen6lpDevice component

The SedonaJen6lpDevice is the device-level component representing a wireless Jennic-based device, and is a direct child of a SedonaJen6lpNetwork. It contains properties and slots typical to most driver's device components—see "Common device components" in the *Drivers Guide* for general information on device component status properties, health, and alarm source info.

No special views are available on the SedonaJen6lpDevice, apart from the standard ones (Property Sheet, Wire Sheet, Category Sheet, Slot Sheet, Link Sheet). However, it has a standard **`Points`** device extension with **`Sedona Point Manager`**, and a **`Chopan Virtual`** gateway—unique to this device type.

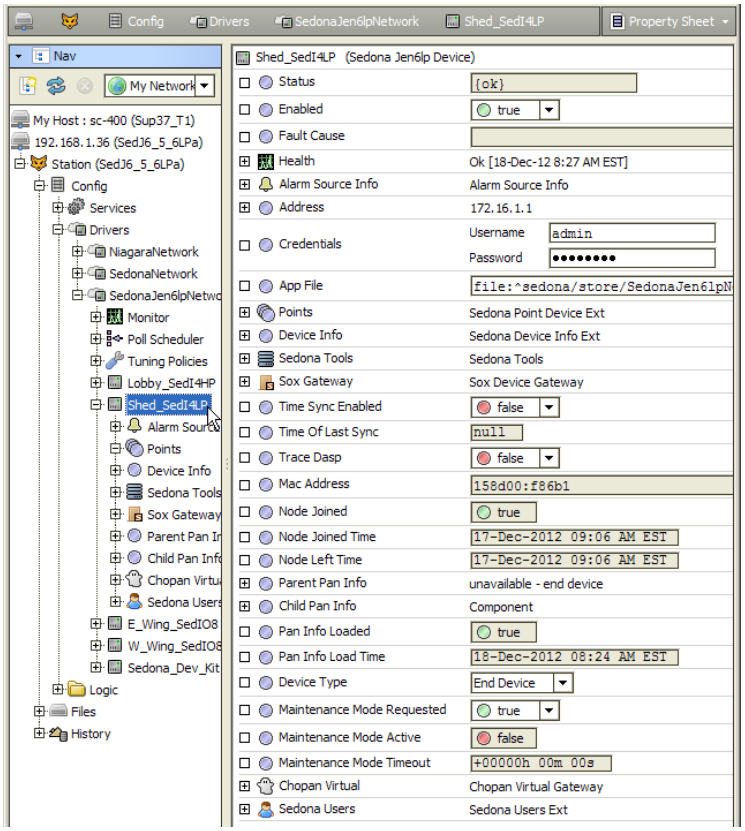
The following sections provide more details:

- "SedonaJen6lpDevice properties" on page 3-36
- "SedonaJen6lpDevice actions" on page 3-39
  - "About maintenance mode" on page 3-39
- "About the Chopan Virtual gateway" on page 3-40

SedonaJen6lpDevice properties

Figure 3-30 shows the property sheet view of a SedonaJen6lpDevice—its default view.

Figure 3-30 *SedonaJen6lpDevice property sheet*



Properties and container slots unique to the SedonaJen6lpDevice include:

- **Address**
  - Ip — Remapped IPv4 private network address of the Jennic-based device, automatically assigned by the coordinator (under the "Address Map" container slot of SedonaJen6lpNetwork).
  - Sox Port — The port the device's Sox service is listening on. Default port is 1876.
  - Chopan Port — The port the device's CHoPAN server is listening on. Default port is 1810.

  Note: *If you know that a Jennic-based device is using a different Sox port and/or Chopan port, change it to match in the address property.*

- **Credentials**
  The credentials used to authenticate to the device in a Sox connection.
- **App File**
  File ord to an associated Sedona app file (typically `.sax` file) for the device, residing in the file space of the station. Initially, there is no associated app file (device is "unassociated"). However, this field becomes populated in a number of ways. For example, when you Get the app (or Put an app) using **`Sedona Tools`** under the device, the app file is populated. Or, you can "directly" associate an app. For related details, see "Sedona device Association" on page 3-20.
- **`Points`**
  The single device extension, with a default **`Sedona Point Manager`** view. For related details, see "About the Sedona Point Manager" on page 3-47.
- **`Device Info`**
  The SedonaDeviceInfoExt extension contains read-only properties about a device's Sedona platform type. For more details, see "Sedona Device Info Ext" on page 3-14.
- **`Sedona Tools`**
  For Sedona provisioning tools for this device. See "Sedona device "tools" views" on page 3-15.
  Note: *Before using these tools, installation of "Sedona environment files" are required on the Niagara host (e.g. JACE) running this station, using a Niagara platform connection. This is also required for "Sox Gateway" usage. For details, see "Sedona environment management" on page B-1.*
  For details on the various Sedona Tools, refer to the *Sedona Framework TXS Sedona Tools Guide*.
- **`Sox Gateway`**
  Provides a "gateway" into that device's Sedona app. For details, see "Sox Gateway" on page 3-16.
- **Time Sync Enabled**
  If enabled, allows for periodic (automatic) time synchronization messages to be sent from the Niagara station to this device. The default value is false. For related details, see "Niagara-to-Sedona device time synchronization" on page 3-18.
- **Trace Dasp**
  If enabled, *and* you have station logSetup (in spy) set to "trace" for `SedonaJen6lpNetwork.sox`, this directs extra device-specific messaging of the Sox traffic over the DASP (Datagram Authenticated Session Protocol) to the station's output (visible in platform Application Director).
- **Mac Address**
  Read-only. The unique 802.15.4 MAC address of the device, in hexadecimal notation without leading zeroes, for example: `158d00:9b017`.
- **Node Joined**
  Read-only. Indicates if the node has joined the Jennic wireless network.
- **Node Joined Time**
  Read-only. Timestamp of when the node last joined the Jennic wireless network.
- **Node Left Time**
  Read-only. Timestamp of when the node last left the Jennic wireless network.
- **Parent Pan Info**
  Container SedonaJen6lpNeighborEntry component for the *single parent node* for the device, *`802.15.4MacAddress`* `LQI` *`n`* (for example, `00158d00_0009b50b LQI 255`)
  Contains read-only JenNet PAN info about the parent node, including:
  - Is Sleeping End Device — Always `false`. (Whether device is a battery powered (hibernating) end device type (`true`) or a continuously powered device (`false`).)
  - Link Quality — Value between 0-255, representing strength of last packet received from this node. Values over 60 represent a good link.
  - Packets Lost — Number of packets sent to this node for which an ack(nowledgement) was not received. Note an ack is not expected on all packets.
  - Packets Sent — Number of packets sent to node from this device.
  - Packets Received — Number of packets device has received from this node.
  - Mac154 — The unique 802.15.4 MAC address of this node, in hexadecimal notation without leading zeroes, for example: `158d00:9b50b`
  - Stale — Typically false, else true if this node entry is stale, meaning the device represented is no longer the parent. For example, if a device resets and rejoins the JenNet network, it may have a new parent (the coordinator or another router node). In this case, this Stale values shows true.
- **Child Pan Info**
  Container for one or more dynamically-added SedonaJen6lpNeighborEntry child components, if applicable. Each contains read-only JenNet PAN info about a direct child node.
  Note: *If the device has no child nodes, this container is empty.*

- MAC*802.15.4MacAddress* LQI *n* (for example, `MAC00158d00_0009b017 LQI 186`)
  - Is Sleeping End Device — Whether device is a battery powered (hibernating) end device type (`true`) or a continuously powered device (`false`).
  - Link Quality — Value between 0-255, representing strength of last packet received from this node. Values over 60 represent a good link.
  - Packets Lost — Number of packets sent to node for which an ack(nowledgement) was not received. Note an ack is not expected on all packets.
  - Packets Sent — Number of packets sent to node from this device.
  - Packets Received — Number of packets this device has received from node.
  - Mac154 — The unique 802.15.4 MAC address of the node, in hexadecimal notation without leading zeroes, for example: `158d00:9b017`
  - Stale — Typically false, else true if this node entry is stale, meaning the node represented is no longer a child. For example, if a device resets and rejoins the JenNet network, it may have a new parent (say the coordinator or another router node). In this case, this Stale values shows true.

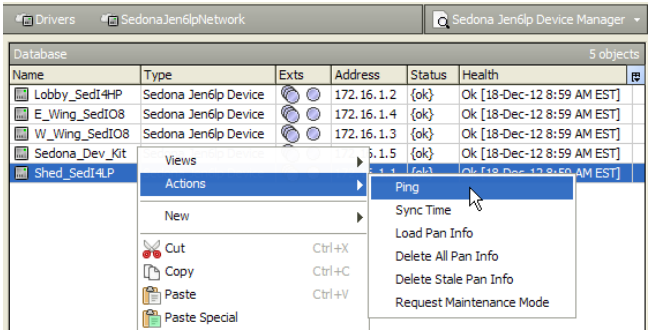
- **Pan Info Loaded**
  Read-only indication if JenNet PAN info for this device is available (`true`) or if it has been deleted (`false`), perhaps from a device action.
- **Pan Info Load Time**
  Read-only timestamp of when PAN info last loaded, say from a device action, or from the ongoing PAN info poller mechanism (from coordinator/SedonaJen6lpNetwork).
- **Device Type**
  Configuration property that specifies whether the Jennic-based device is either "router capable" (Router), or reduced function "end device" (End Device—perhaps even a hibernating device), or simply "Unknown". The default is Unknown.
  Note: *Set this from the default (Unknown) to one of the two other selections (Router or End Device), depending on the Jennic capability of the device. This is particularly important to allow "PAN info" polling and related data to be used in the Pan Sheet view of the SedonaJen6lpNetwork.*
- **Maintenance Mode Requested**
  (Applies to hibernating devices only) This provides indication if a "maintenance mode request" has been given on this device (`true`) or not (`false`). The "Request Maintenance Mode" *action* on the device is typically how a request is given. This value goes to true when the action is invoked, then to false when maintenance mode becomes active—when in Workbench a modal popup "maintenance mode active" appears. See "SedonaJen6lpDevice actions" on page 3-39.
  Note: *Maintenance mode only applies to a device that hibernates (say a battery-powered device), and requires Chopan usage in the device's app. If the device does not hibernate, you can safely hide this property, as well as the other two Maintenance Mode properties (and also the "Request Maintenance Mode" action), working from the SedonaJen6lpDevice's slot sheet.*
  Currently, Sedona Framework support for hibernating devices is not widely available, so this is another possible reason to hide these Maintenance Mode properties and action.
- **Maintenance Mode Active**
  (Applies to hibernating devices only) Provides indication if "maintenance mode" is currently active on this device (`true`) or not (`false`). It becomes true only after the modal "maintenance mode active" popup appears in Workbench, and returns to false when the maintenance mode time out period (configured in the device's Sedona Framework app) expires. During this period, the device is not hibernating, which enables it to accept a Sox connection from Workbench. Also see the Note: on the "Maintenance Mode Requested" property.
- **Maintenance Mode Timeout**
  (Applies to hibernating devices only) Read-only indication of the current maintenance mode timeout period, as configured in the Sedona Framework app of the device. This specifies the duration of the maintenance mode active period. If a Sox connection is not established during an active period, the device returns to normal hibernation operation, until the next maintenance mode request. Also see the Note: on the "Maintenance Mode Requested" property.
- **`Chopan Virtual`**
  A "virtual gateway" used to configure "Chopan points" in the device's Sedona Framework app, via a virtual "Chopan Network" with "Chopan Devices" and "Chopan Points" in the Niagara station. See "About the Chopan Virtual gateway" on page 3-40.
- **`Sedona Users`**
  This SedonaUsersExt component registers with the station's SedonaUserManagementService (SedonaUserManager), and *internally* stores the mapping of Sedona users and roles for the device (as defined in the **`Sedona User Device Manager`** view of the parent Sedona network). This com-

ponent manages synchronization of those mappings down to the app in the represented Sedona device. For related details, see "Sedona Device User Manager view" on page 3-11.

SedonaJen6lpDevice actions

Figure 3-31 shows the default actions available on a SedonaJen6lpDevice.

Figure 3-31 *SedonaJen6lpDevice actions (default)*



Device actions include the following:

- **Ping**
  Verifies join status with the coordinator (SedonaJen6lpNetwork).
- **Load Pan Info**
  Forces a load of the device's JenNet PAN info into its Parent Pan Info and Child Pan Info properties.
- **Delete Pan Info**
  Deletes all JenNet PAN info from the device's Parent Pan Info and Child Pan Info properties.
- **Delete Stale Pan Info**
  Deletes all JenNet PAN info entries currently marked "stale" (property Stale = true). These entries may exist after the tree structure of the Jennic wireless network changes.
- **Request Maintenance Mode**
  Applies only to hibernating devices (say battery-powered)—to provide the ability to make a Sox connection to it from Workbench. For more details, see "About maintenance mode".
  Note: *If a non-hibernating device, this action has no purpose and may be safely hidden. As currently Sedona Framework support for hibernating devices is not widely distributed, this is yet another reason to hide this action.*

About maintenance mode

Note: *Maintenance mode is a routine that applies to hibernating devices only. At the time of this document, Sedona Framework support for hibernating devices is not widely available. However, NiagaraAX support for such devices, including "Maintenance Mode" operation, is included in the SedonaJen6lpNetwork driver.*

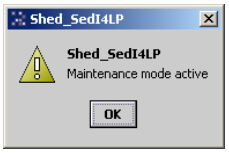
Most hibernating devices, typically battery-powered devices, remain in a low-power or "hibernating" state. During this state, the Sedona VM and RF communications are disabled to conserve battery life. Consequentially, Workbench cannot make a Sox connection to the device.

In order to allow a Sox connection from Workbench to a device that hibernates, its Sedona Framework app is configured with a "ChopanNetwork" *client* that periodically queries the coordinator (JACE) to see if it should enter a *maintenance mode*.

This request is typically done using the "Request Maintenance Mode" action on the SedonaJen6lpDevice that represents the hibernating device. Once invoked, upon the next "maintenance mode check" message from the device (as part of its Chopan setup), the device sees that flag set, and then suspends hibernation for a "maintenance mode timeout" period.

At that time, in Workbench a modal "Maintenance mode active" popup appears, with the Niagara name of the associated SedonaJen6lpDevice component. See Figure 3-32 for an example popup.

Figure 3-32 *Example maintenance mode popup in Workbench (for device named* `Shed_SedI4LP`*)*



The popup signals the start of the "maintenance mode active time". During this period, before the timeout, you can use Workbench to open a Sox connection to that device (typically a Sox tunnel connection). If a Workbench Sox connection is not made in that time period, the device returns to hibernating, that is, resumes its normal operation.

Three related properties of a SedonaJen6lpDevice indicate maintenance mode states and the (Sedona configured) maintenance mode timeout duration. See "SedonaJen6lpDevice properties" on page 3-36.

About the Chopan Virtual gateway

The Chopan Virtual gateway is used to configure "Chopan points" in the device's Sedona Framework app, via a virtual "Chopan Network" with "Chopan Devices" and "Chopan Points" in the Niagara station. The resulting Chopan components added in the device's app act as the Chopan *client interface* to other devices' "Chopan servers" (including the one for the station, in the SedonaJen6lpNetwork).

To support virtual gateway access from Niagara, the Jennic-based device requires the `chopan` kit installed, and its Sedona Framework app configured with components from it.

- If a non-hibernating (typically continuously-powered) device, Chopan usage is optional. However, Chopan as a "comm type" for normal Sedona proxy points offers bandwidth efficiencies. Also, via Chopan points, a device can exchange data directly with another networked Jennic-based device.
- Such configuration is *required* if a hibernating[1] device (typically a battery-powered device), as Chopan points must be used instead of normal Sedona proxy points to exchange data with the station. Chopan is also used in the "maintenance mode" mechanism for a hibernating device.

A brief overview on Chopan points follows in this document, in the following sections:

- Required configuration in the Jennic-based device
- Walk-through using Chopan Virtual gateway
- Additional Chopan usage topics

Note: *For complete details see the* Sedona Framework Chopan Usage *document.*

Required configuration in the Jennic-based device

To support Chopan, the Jennic-based device requires the following:

- The chopan kit installed.
- To function only as a CHoPAN client (hibernating device support), its Sedona Framework app must configured with the following:
  - ChopanService (from chopan kit) added to its Service container.
  - ChopanNetwork (from chopan kit) added to its Service container.

  ChopanDevices and ChopanPoints can then be added to its app using the special manager views under the Chopan Virtual gateway of the SedonaJen6lpDevice that represent it in the station.
- To function as a CHoPAN server, its Sedona Framework app must configured with the:
  - ChopanService (if not already present, from chopan kit) added to its Service container.
  - ChopanServlet (if not already present, from chopan kit) added under the ChopanService.

  Chopan server configuration is typical for most Jennic-based devices (non-hibernating), to allow Sedona proxy point updates using a tuning policy with "Chopan" as the comm type. See "Chopan comm type considerations" on page 3-27. Additionally, the device can be configured for Chopan client operation too, if direct access to data in some other networked Jennic-based device is needed.

Walk-through using Chopan Virtual gateway

This walk-through describes using CHoPAN to exchange data between a hibernating Jennic-based device and the JACE station, where the "Chopan Server" of the SedonaJen6lpNetwork is enabled (default), and the hibernating device has its app configured as a CHoPAN client and server (as described in section "Required configuration in the Jennic-based device" on page 3-40). Note all of the following *replaces* Sedona proxy point usage for this hibernating Jennic-based device.

1. Currently, Sedona Framework support for hibernating devices is not widely available. Chopan Virtual gateway usage without this support should be minimal, as it is necessary only if a Jennic-based device needs to request data directly from another networked Jennic-based device.

Example Chopan point setup using Chopan Virtual gateway

Before using the Jen6lpDevice's Chopan Virtual gateway, it would be best to have "target" writable Niagara points ready to receive read-only values from the hibernating Jennic-based device. In the Niagara jen6lp palette, there is a folder named "ChopanTargetPoints", with two conveniently "trimmed" writable points:

- ChBoolTarget — a normal BooleanWritable, but with *all actions and all inputs except* "in10" *hidden*.
- ChFltTarget — a normal NumericWritable, but with *all actions and all inputs except* "in10" *hidden*.

Actions and other input slots are hidden to avoid confusion trying to control (write to) this read-only value from Sedona, say if dragging this control point on a Px page. Typical usage of these points is to receive some value like a temperature, or an equipment state from the device's Sedona Framework app.

As a best practice, and for database portability/replication, it is recommended that you place such Chopan target points in a *folder under the SedonaJen6lpDevice* that represents the hibernating device. Note that the standard "Points" folder is not the best choice—as these are *not* proxy points, and for this reason do not appear in the Sedona Point Manager view.

Figure 3-33 *Adding ChopanTargetPoints for read-only values, in folder under device component*

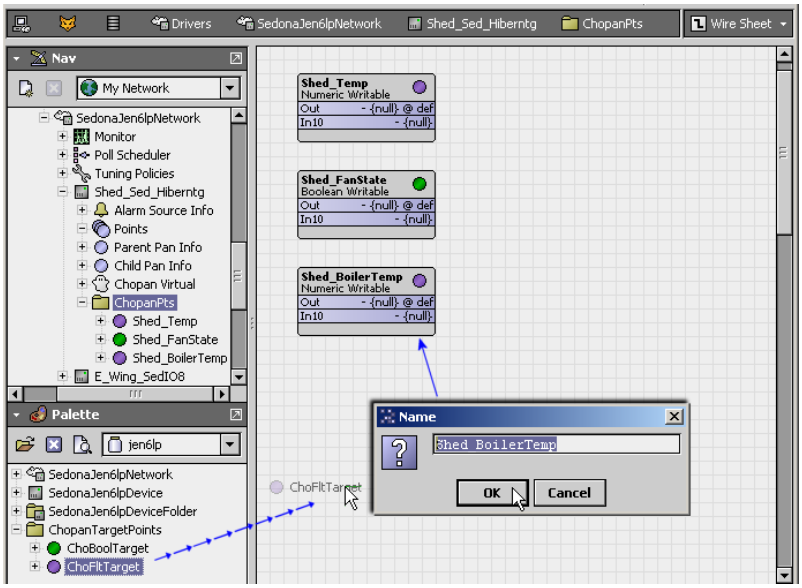


Figure 3-33 above shows three such points copied from the jen6lp palette into a "ChopanPts" folder under the SedonaJen6lpDevice (named "Shed_Sed_Hibernatng").

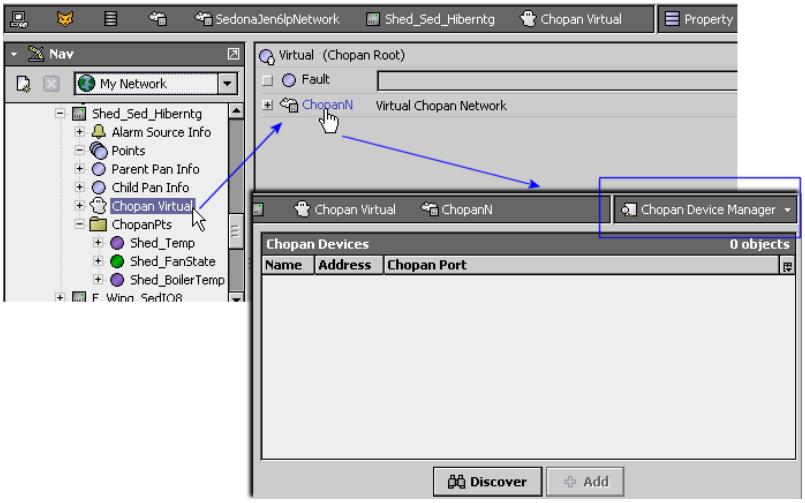
Now, onto using the gateway.

Note: *In the following steps, virtual components must be clicked (versus expanded) to "enter" them.*

Step 4 The **`Chopan Virtual`** gateway is clicked, and it soon expands to reveal a **`ChopanN`** component (Virtual Chopan Network) in its property sheet.

Step 5 The ChopanN component is clicked, and the view changes to the **`Chopan Device Manager`** for that device. See Figure 3-34.

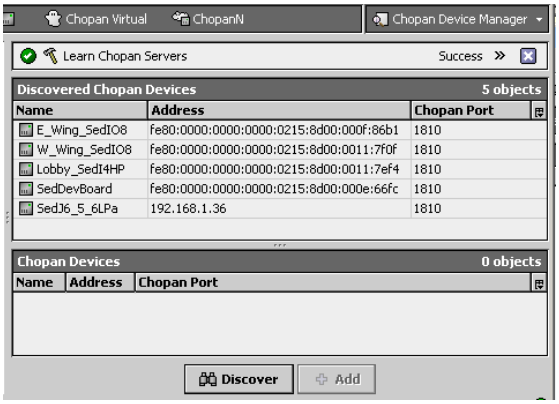
Figure 3-34 *Chopan Device Manager is view on the ChopanN (Virtual Chopan Network)*



As shown in Figure 3-34, initially the **`Chopan Device Manager`** is empty—a Discover is needed to find other devices configured with a Chopan server.

Step 6 The **`Discover`** button is clicked, launching a **`Learn Chopan Servers`** job.

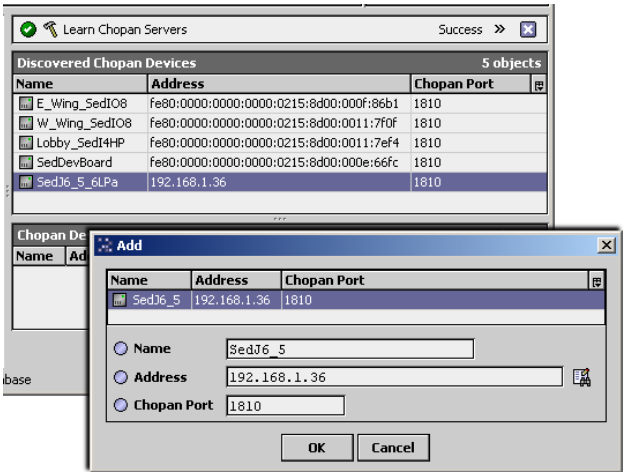
Figure 3-35 *Discovered Chopan servers in Chopan Device Manager*



As shown in Figure 3-35, five devices were found, of which one is the JACE station (“SedJ6_5_6LPa”). In most cases, this is the device of primary interest—and the one to be added here.

Step 7 The discovered JACE station is double-clicked to bring up the Add dialog, shown below.

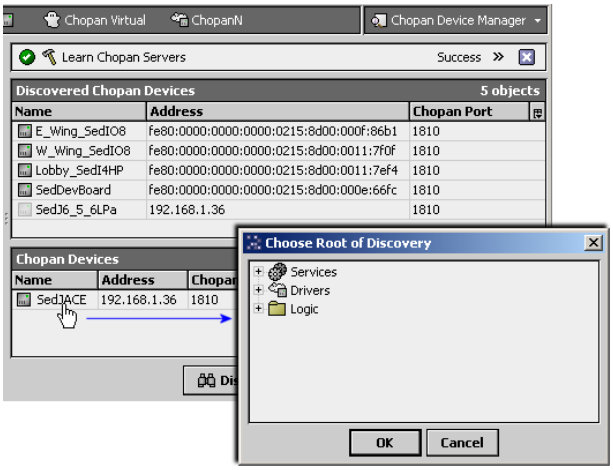
Figure 3-36 *Adding Chopan Device that represents the JACE station*



Notice that Name is truncated at 7 characters—the maximum number of characters for naming any Sedona component. In this case, the name is changed from “SedJ6_5” to “SedJACE”, and then added (**`OK`**).

Step 8 The added SedJACE Chopan Device is then double-clicked.

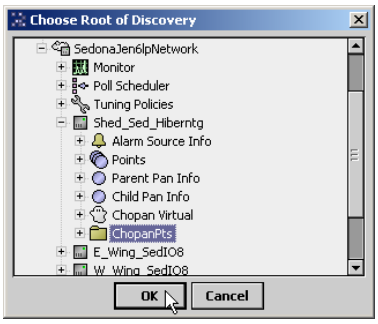
Figure 3-37 *Point discovery of Chopan Device brings up*



As shown in Figure 3-37, this brings up a "Choose Root of Discovery" dialog, reflecting the contents of the **`Config`** node of the JACE station, in an expandable tree.

Step 9 The tree is expanded to find the previously added "ChopanTargetPoints" for this device.

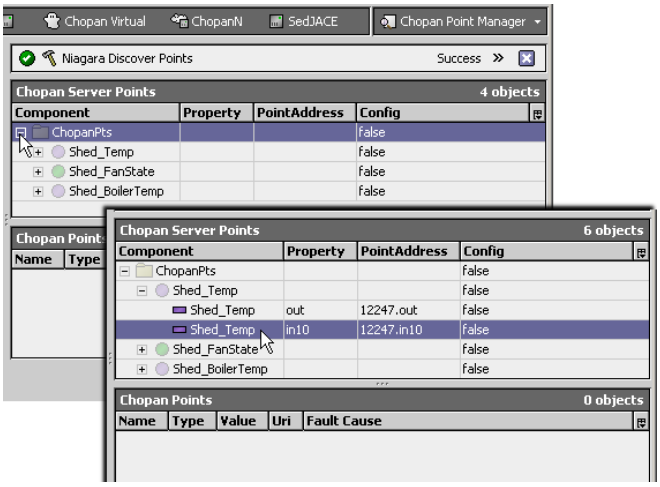
Figure 3-38 *Drivers leaf expanded to find Chopan target points for this device*



As shown in Figure 3-38, the previously-created folder "ChopanPts" is selected as the root (**`OK`**). The view changes to the **`Chopan Point Manager`** for the (JACE) device.

Step 10 In the Discovered pane, the folder is expanded to reveal the target points, then available properties.

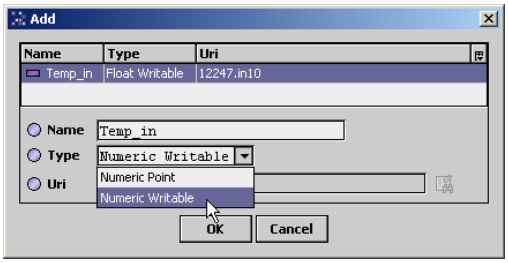
Figure 3-39 *Expanding root folder in Discover to find properties of Chopan target points.*



The "in10" property is double-clicked to bring up the **`Add`** dialog.

Step 11 The **Add** dialog defaults to a NumericPoint, with truncated name (in this case, "Shed_Te").

Figure 3-40 *Add dialog for Chopan target point, changing to writable type point*

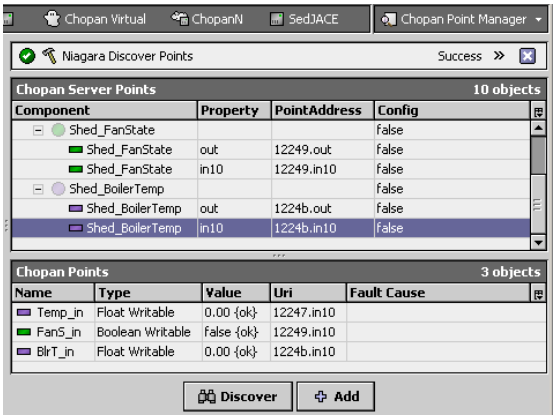


As shown in Figure 3-40, name is edited to "Temp_in", and point type *changed* to NumericWritable, and the point is added (**OK**). The actual Sedona component added is a `ChoFltWr` (Chopan Float Writable).

This needs to be a writable point, because the source Sedona temperature component will be linked to it in the device's Sedona Framework app.

Step 12 The same method is used to add two other Chopan points in the device, also writable (Figure 3-41).

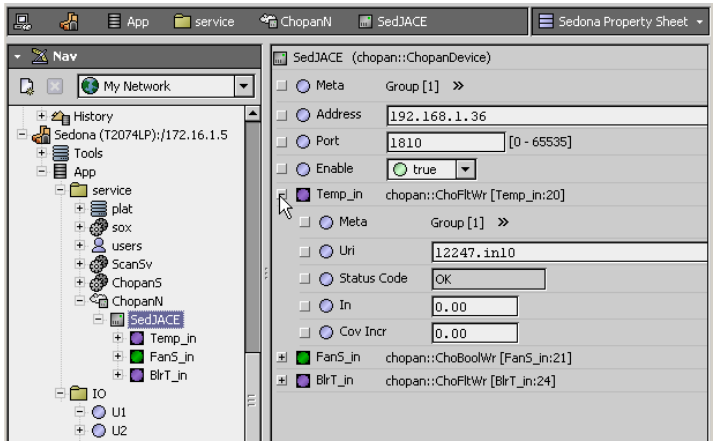
Figure 3-41 *Three Chopan points now in the Chopan Point Manager view*



Note that these three Chopan point components, as well as the parent Chopan device component, *now exist in the Sedona Framework app* of the hibernating device. To make them operational, a Sox Workbench connection must be made to this device, such that the source Sedona components can be linked to them.

Step 13 A "Maintenance Mode Request" action is invoked on the SedonaJen6lpDevice for this device, and following the "Maintenance Mode Active" popup, a tunneled Sox Workbench connection is made to it.

Figure 3-42 *Workbench Sox connection made to hibernating device, showing added Chopan points*



As shown in Figure 3-42, the Chopan points in the device's app are under the "ChopanN", "*ChopanDeviceName*" node under the **service** folder (and must remain there). One way to link the source Sedona components to them is using the Nav tree, and right-click "Link Mark" and "Link From" menu options.

Step 14 Source Sedona components are linked to the Chopan points, using “Link Mark” and “Link From” methods.

Figure 3-43 *Linking source Sedona components to Chopan writable points*

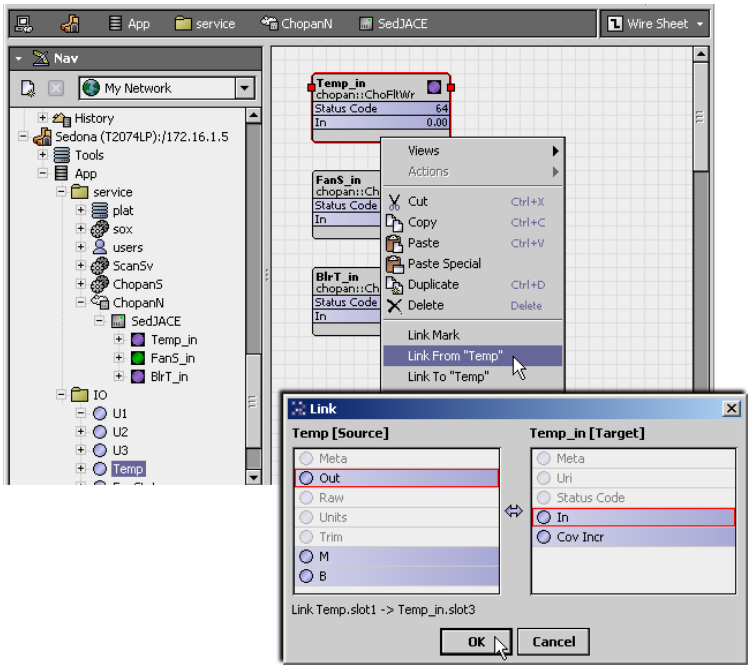


Figure 3-43 shows the **`Link`** dialog for the “Temp” (temperature) source component to the “Temp_in” writable Chopan point, where the “In” slot is used.

Step 15 After making source links to all the writable Chopan points, the app must be saved.

Figure 3-44 *Saving Sedona Framework app of the device after linking in Chopan points*

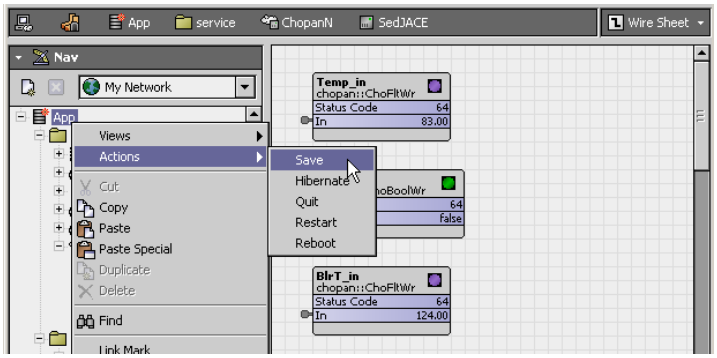
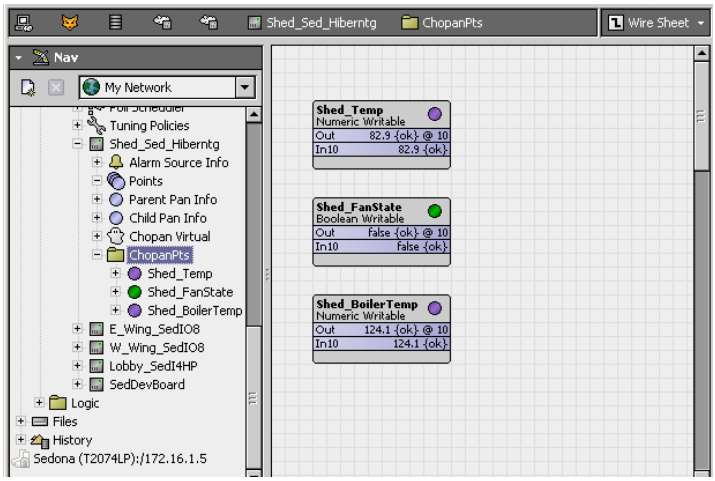


Figure 3-44 shows the right-click **`Save`** command invoked on the App node. The Workbench Sox session to the hibernating device is now closed, and the target Chopan points are checked in the JACE station.

Step 16 Chopan target points in the station are now receiving updates from the hibernating device, via CHoPAN, as shown in Figure 3-45.

Figure 3-45 *Chopan target points in station, with values received via CHoPAN.*



Control points shown in Figure 3-45 still need point facets assigned, and whatever linkage into station control logic, and/or bindings to Px page(s).

Additional Chopan usage topics

The following topics are not in the previous Chopan point "walk-through", but are mentioned briefly below. For more details on Chopan topics, refer to the *Sedona Framework Chopan Usage* document.

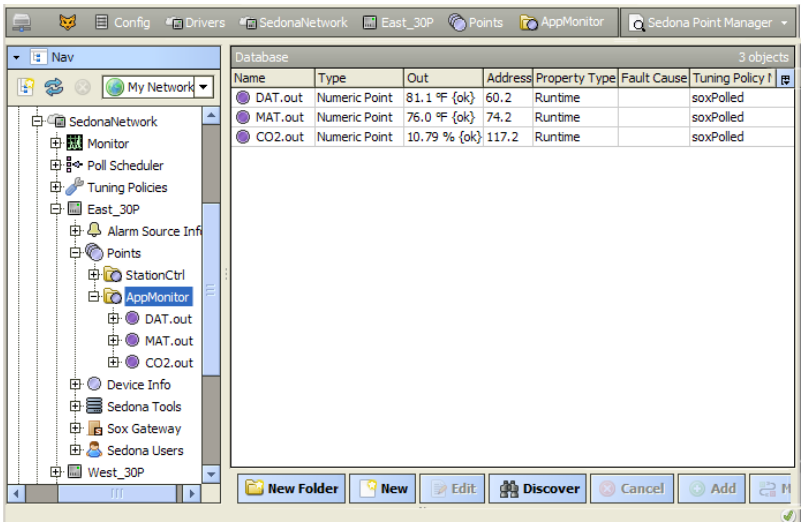
- When the Chopan Server of the SedonaJen6lpNetwork is enabled, this automatically results in the creation of a station user named "CHoPAN". This is necessary because CHoPAN is an unathenticated protocol, where CHoPAN servers do not authenticate or prevent and reads or writes to local components. Niagara stations allow reads of any appropriate component, but *writes* to station components are checked for authorization (as part of this CHoPAN user scheme). Therefore, any incoming CHoPAN requests that result in station database modification are checked against the permissions assigned to the CHoPAN user. In other words, the Niagara component being written must be assigned to a category for which the CHoPAN user has write privileges.
- Any Sedona Framework Jennic-based device can be configured in its app to support "service pin notifications", using components from the `chopan` kit. The app must contain the "ChopanS" (ChopanService) and "ChopanN" (ChopanNetwork) components, with others optional. See the "Service Pin support" section in the *Sedona Framework Chopan Usage* document.
- Chopan points have a *status code* slot showing a numerical value that reflects the response of the latest request. Codes also have a descriptive "Name" that appear in the Chopan Point Manager view of a Chopan server device (under the SedonaJen6lpDevice's **`Chopan Virtual`** gateway), which is useful for troubleshooting purposes. Refer to the "Chopan Diagnostics" section in the *Sedona Framework Chopan Usage* document for status code names, numerical values, and their meanings.

About the Sedona Point Manager

The **Sedona Point Manager** is the default view on the **Points** device extension (or a Points folder under that) for any type of Sedona device component (SedonaDevice or SedonaJen6lpDevice), and operates the same way for either type of device.

Note: *For any* hibernating *SedonaJen6lpDevice (typically a battery-powered device), use of this view to create Sedona proxy points* is not recommended*. Instead, use the device's Chopan Virtual gateway to integrate data with the station. See "About the Chopan Virtual gateway" on page 3-40 for an overview.*

Figure 3-46 *Sedona Point Manager example*



The **Sedona Point Manager** is *somewhat* similar to the point managers in other NiagaraAX drivers that feature points folders and online point discovery (Learn mode). For general information on point managers, refer to the *Drivers Guide* sections "Points New Folder and New" and "About Point Discover, Add and Match (Learn Process)".

The following sections provide more details about the **Sedona Point Manager**:

Unique things about the Sedona Point Manager

The following things are unique to the **Sedona Point Manager**:

- **Point discovery can be online or offline**
  Starting in Sedona TXS-1.2, providing a Sedona device has an app file "associated" with it, you can perform either an online or *offline* point discovery to add Sedona proxy points to the station database. For the online procedure, see "Create Sedona proxy points (and action points)" on page 2-13. For details on "device to app association", see "Sedona device Association" on page 3-20. For offline discovery details, refer to the *Sedona Framework TXS Offline Engineering Guide*.
- **Manifests required on the** JACE
  For proxy point support of any Sedona Framework device, the host (e.g. JACE) running the station with the Sedona Framework network requires at a minimum, the *manifest* file for each kit installed in the device. Otherwise, a "Sedona Discover Points" job will fail, and an error popup similar to Figure 3-47 below appears in Workbench.

Figure 3-47 *Example error popup on Sedona Discover Points job (missing manifest)*

To fix this, we recommend you install all the necessary Sedona environment files on the JACE to support networked Sedona devices, which include the appropriate kit manifest files. The recommended way to do this is with a *platform connection* to the JACE, using the **`Sedona Environment Manager`** view. Then rerun the point discovery.

For related procedures, see sections "JACE workflow" on page 1-2 and "Installing Sedona environment files in a JACE" on page 1-3 in the "Sedona Framework Network Driver Installation" section.

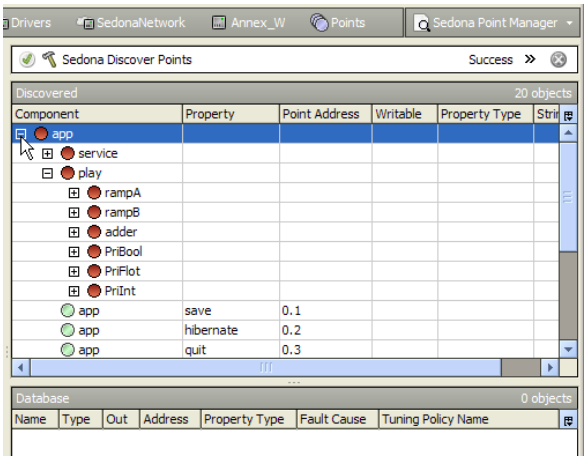
Note: *If a platform connection in unavailable, you could obtain the appropriate kit manifest file(s), and then transfer them to the JACE using the* **`Manifest Manager`** *view on either Sedona Framework network (SedonaNetwork, SedonaJen6lpNetwork). See "Run the Manifest Manager to ensure kit manifests are loaded" on page 2-12. Then rerun the point discovery.*

- **Action points available**
  In addition to Sedona proxy points, the Sedona Point Manager lets you add "action points", which are unique to Sedona Framework devices. Each action point is a primitive component that represents a single action on the source Sedona component. Action points do not read data values nor accept point extensions like alarm, history, and so on—and have only a single property (Address). An action point makes that same action available in Niagara, named using the action slot name on the parent Sedona component. If needed, you can add a display name for that action, working from the slot sheet of the action point. For more details, see"About Sedona action points" on page 3-55.
- **Default point type may need changing**
  Each Sedona proxy point represents a single property of a Sedona component in the device's Sedona Framework app. Knowledge of the device's app is often required when making these decisions. Variations between data types in Sedona and Niagara should be understood for type selection of proxy points. There is no "enum" (multistate) data type[1] in Sedona, however, "enum" point types default for any selected property that uses an integer data type (int, long). Sometimes this may be appropriate. Other times, you may wish to change type when adding, for example from "Enum" to "Numeric" for an integer property representing a count.
  For more details, see "About Sedona proxy points" on page 3-52, and "Sedona and Niagara data types and null notes" on page 3-53.

Sedona Point Manager "Discovered" notes

By default, the "Discovered" table for Sedona points shows a single, red icon, **`app`** component at top.

Figure 3-48 *Discovered Sedona components, folders, and child properties and actions*



1. The boolean data type in Sedona is effectively multistate, as it can be either false, true, or null. This is the only multistate instance in Sedona. For more details, see "Booleans as Enums" on page 3-54.

When expanded, other components and folders appear with the same red icon—each is expandable, but not directly selectable. Only child *properties* (, ,) or *actions* ( ) are selectable. Note that Sedona component names cannot be more than 7 characters—as seen in the "Component" column.

Click to expand any component or folder, as shown in Figure 3-49.

Figure 3-49 *Click plus (+) icon beside any discovered component to see all properties*

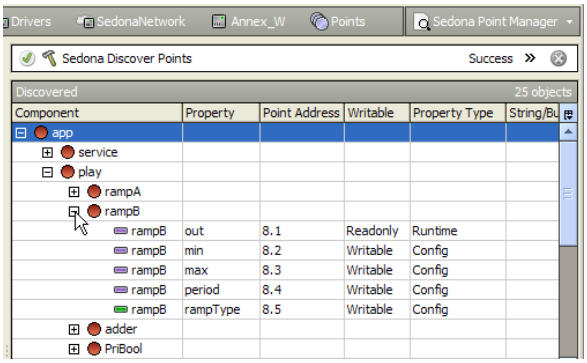


Table columns in the Discovered pane are as follows:

- **Component**
  Name of the parent Sedona component or folder, from 1 to 7 characters maximum.
- **Property**
  Sedona slot name for a property or action—i.e. how it is listed on the component's slot sheet.
- **PointAddress**
  Unique numerical ID of each Sedona component and child slot in the app, in form `compId.slotId`, for example `9.4` for the fourth slot of the ninth component.
- **Writable**
  Whether a slot can be written (`Writable`), or else not (`Readonly`). If adding a proxy point for a writable property, the Add dialog allows selection of either a read-only point or writable point.
  Note: *A writable property may be linked to another source component in the device's Sedona app.*
- **Property Type**
  Whether a slot's value is persisted in an app save (`Config`), or else not (`Runtime`).
- **String/Buf**
  Blank (no value) unless a Sedona property type of `sys::Buf`, where it either shows `Str`, to indicates a null-terminated string value, or else `Buf`, to indicate a string as byte array or other data structure.

As in other point managers, to initiate an add you can double-click or drag-and-drop an item in the lower Database pane (for single items), or hold the Ctrl key down and click (select) multiple items to add with the **Add** button. This produces the **Add** dialog. See "Add dialogs in Sedona Point Manager".

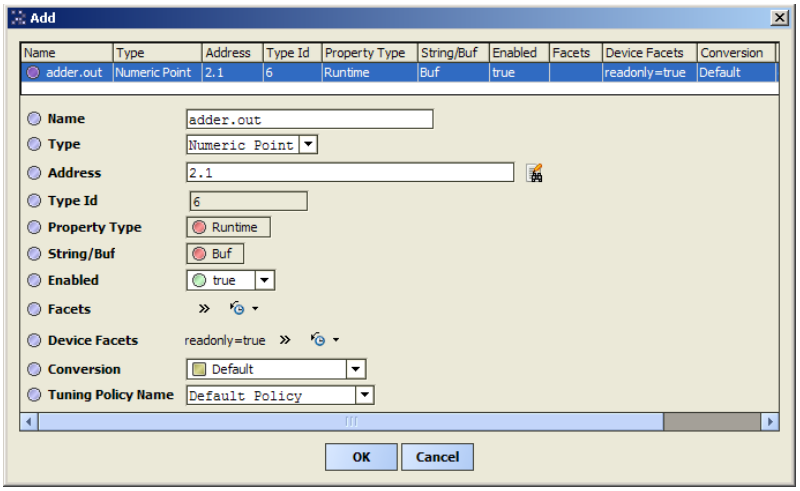
Add dialogs in Sedona Point Manager

Depending if adding property items or action items, the **Add** dialog in the Sedona Point Manager varies.

Sedona proxy point Add dialog

An example **Add** dialog for a Sedona proxy point is shown in Figure 3-50 below.

Figure 3-50 *Proxy point Add dialog in Sedona Point Manager*



Data fields in the **Add** (and **Edit**) dialog for a proxy point contain most properties in the SedonaProxyExt, along with a few others. The fields are as follows:

- **Name**
  Niagara name of the Sedona proxy point, which defaults to Sedona `ComponentName.slotName`, which can be left at default or changed if needed.
- **Type**
  Niagara control point type for the proxy point, for example, Numeric Point, Numeric Writable, Boolean Point, Boolean Writable, and so on.
  Note: *Chose Type before making changes to other properties below; otherwise they become reset. Unlike other editable entries in the Add dialog, you cannot edit Type later in the Edit dialog.*
- **Address**
  Unique numerical ID of the property in the app, in form `compId.slotId`, for example `9.4` for the fourth slot of the ninth component.
  Note: *Automatically determined at proxy point add time. Manual editing is not recommended.*
- **Type Id**
  Read-only integer value, used internally to properly encode write values.
- **Property Type**
  Either or `Runtime` or `Config`.
  - If **`Runtime`**, the property's value is *not saved* to non-volatile Flash memory on an app save.
  - If **`Config`**, an app save preserves the property's value to non-volatile Flash memory.

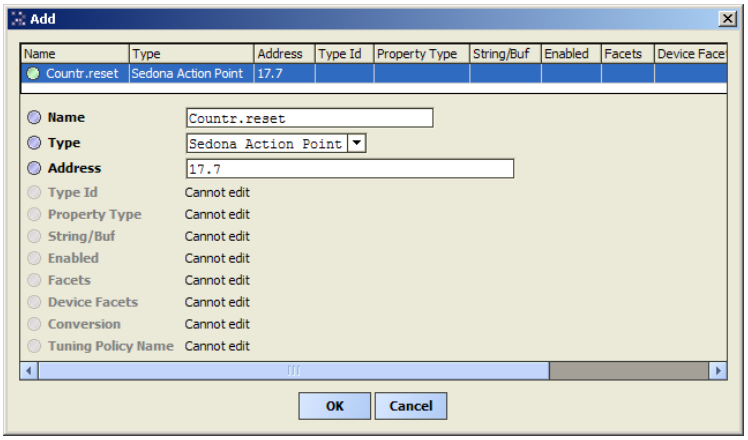
  Often, runtime properties also have a "readonly" flag set—if so, typically the default proxy point type selected in the **Add** dialog a read-only type (BooleanPoint, NumericPoint, etc.).
- **String/Buf**
  Read-only value, either **`Buf`** or **`Str`**. If Buf, the slot's value may be interpreted as a byte array or other data structure. If Str, the slot's value may be interpreted as a null-terminated string. Applicable only if the Sedona property is type `sys::Buf`.
- **Enabled**
  Either **`true`** (default) or **`false`**, to enable/disable the proxy point. If set to false, the point's status is disabled, and any value polling is stopped.
- **Facets**
  Point facets for the proxy point value, which are typically edited from defaults.
- **Device Facets**
  Device facets for the source slot's value, if any.
- **Conversion**
  Specifies the conversion used between the "read value" (in Device Facets) and the parent point's output (in selected point facets). Typically left at the "Default" selection.
- **Tuning Policy Name**
  Specifies which of the network's (SedonaNetwork or SedonaJen6lpNetwork) tuning policies should be used for reads and (if applicable) writes to this proxy point.
  For related details see:
  - "SedonaNetwork tuning policy notes" on page 3-7
  - "SedonaJen6lpNetwork tuning policy notes" on page 3-27

Once added to the station, you can edit any of these fields if needed—with the exception of **`Type`**.

Sedona action point Add dialog

An example **Add** dialog for a Sedona action point is shown in Figure 3-50 below.

Figure 3-51 *Action point Add dialog in Sedona Point Manager*



There are only a few fields in the **Add** (and **Edit**) dialog for an action point, described as follows:

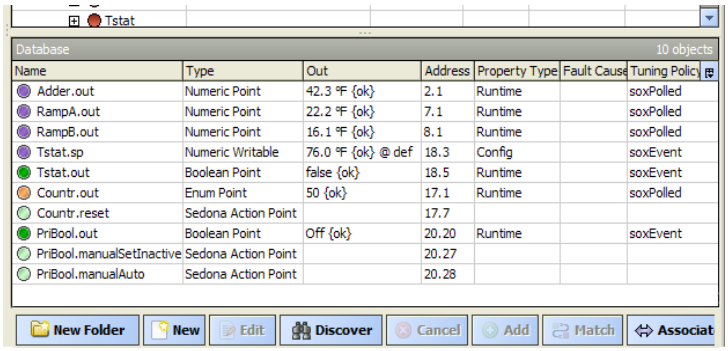
- **Name**
  Niagara name of the Sedona action point, which defaults to Sedona *ComponentName.slotName*. This can be left at default or changed if needed.
- **Type**
  Always **Sedona Action Point**, and not editable after adding point.
- **Address**
  Unique numerical ID of the action in the app, in form *compId.slotId*, for example 17.7 for the seventh slot of the 17th component.
  Note: *Automatically determined at action point add time. Manual editing is not recommended.*

For more details on action points, see "About Sedona action points" on page 3-55.

Sedona Point Manager "Database" notes

The "Database" table lists existing Sedona proxy points and action points, where each appears as a row in the table. Each proxy point represents one property of a Sedona component in that device; each action point represents a single action on a Sedona component (Figure 3-52).

Figure 3-52 *Database shows Sedona proxy points and Sedona action points*



You can resort items by clicking on any column header. This may be useful to sort by type, or perhaps by name.

Also, if you created point folders under a device's Points container, you can see all proxy points and action points in the device from the main (root) Points Manager using "All Descendants" tool. For details, see the section "All Descendants" in the *Drivers Guide*.

The following sections provide more details about the Database pane in the Sedona Point Manager:

- Database point table columns
- Modifying the Database table

Database point table columns

Note: *Most data columns apply to proxy points, where only columns Name, Type, and Address are also applicable to Sedona action points.*

By default, the following columns appear in the **`Database`** table:

- **Name**
  Niagara name of the Sedona proxy point or action point. By default, names use the format: *`ComponentName.slotName`*. Names can be edited or left at default. If displayNames are assigned to proxy points and action points (from the slot sheet of **`Points`**), they display here instead.
- **Type**
  Niagara control point type for proxy points (**`Boolean Point`**, **`Numeric Writable`**, and so on) or if an action point, **`Sedona Action Point`**.
- **Out**
  The current read property value, reflecting point facets, along with point status.
- **Address**
  Unique numerical ID of the slot in the device's Sedona Framework app, in form *`compId.slotId`*, for example `9.4` for the fourth slot of the ninth component.
- **Type Id**
  Read-only integer value, used internally to properly encode write values to the property.
- **Property Type**
  Whether the target Sedona property value is saved to the device's non-volatile Flash memory upon a save to its Sedona Framework app. If the value is saved, "Config" appears; otherwise it is "Runtime".
- **Fault Cause**
  If a proxy point has a fault status, the fault cause text string explains why.
- **Tuning Policy**
  The assigned tuning policy for the proxy point, by name of the network's tuning policy.

Note: *You can also modify columns shown, see "Modifying the Database table" on page 3-52.*

Modifying the Database table

You can modify which data columns appear in the Sedona Point Manager database table. For the options menu, simply click the small "table options" control in the upper right corner. See the section "Manager table features" in the *Drivers Guide* for general details.

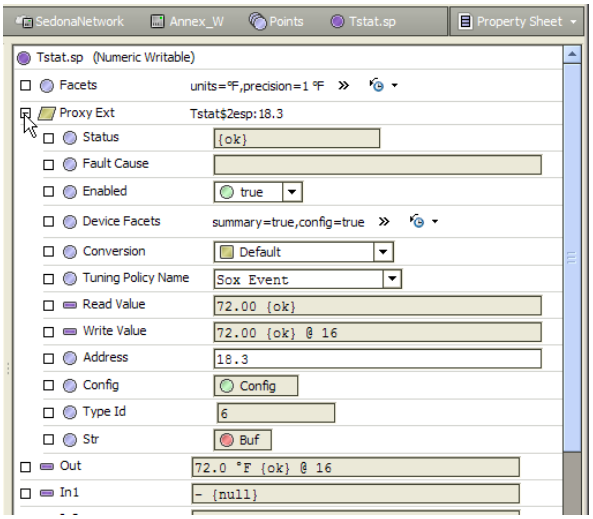
Non-default selections for data columns in the Sedona Point Manager Database table include various properties of both the parent proxy point and the SedonaProxyExt, and are the following:

- Path — Station path to the proxy point.
- Type Id — Numerical identifier for the type of Sedona property.
- String/Buf — Whether target string property is considered "Buf" or "Str" (if applicable).
- Enabled — Whether the Niagara proxy point is currently enabled for communications.
- Facets — Facets in use by the parent proxy point.
- Device Facets — Learned facets of source Sedona property, if any.
- Conversion — Conversion used between device facets and point facets (typically "Default").
- Read Value — Last value read from device, expressed in device facets.
- Write Value — (Applies to writable types only) Last value written, using device facets.

About Sedona proxy points

Sedona proxy points are similar to other driver's proxy points, meaning they are based on the same core Niagara control points, with the SedonaProxyExt. The SedonaProxyExt uses the same core properties of most driver's proxy points, such as Status, Fault Cause, Enabled, and so on. For general information, see the section "ProxyExt properties" in the *Drivers Guide*.

Figure 3-53 *Example SedonaProxyExt in property sheet*



The following SedonaProxyExt properties are of particular importance:

- **Tuning Policy Name**
  Specifies which of the network's (SedonaNetwork or SedonaJen6lpNetwork) tuning policies should be used for reads and (if applicable) writes to this proxy point.
  For related details see:
  - "SedonaNetwork tuning policy notes" on page 3-7
  - "SedonaJen6lpNetwork tuning policy notes" on page 3-27
- **Read Value**
  (read only) Last value read from the device, expressed in device facets.
- **Write Value**
  (read only) Applies if writable point only. Last value written, using device facets.
- **Address**
  Unique numerical ID of the property in the app, in form `compId.slotId`, for example `9.4` for the fourth slot of the ninth component.
  Note: *Automatically determined at proxy point add time. Manual editing is not recommended.*

For details on other SedonaProxyExt properties, see "Sedona proxy point Add dialog" on page 3-49.

Note: *Sedona proxy points are modeled using standard Niagara data types. For details, see "Sedona property data types to default Sedona proxy point types" on page 3-53.*

For a procedure on adding points, see "Create Sedona proxy points (and action points)" on page 2-13.

Sedona and Niagara data types and null notes

When comparing data types in Sedona and Niagara, note that properties in Sedona components use one of several primitive data types, including "bool" (boolean value), "int" (signed 32-bit integer), "float" (32-bit floating point), and several others. For complete details, see the documentation on the Sedona Framework org website, for example at http://www.sedonadev.org/doc/primitives.html.

Niagara models this data with Sedona proxy points, using standard control points based on "status value" data types (statusBoolean, statusNumeric, statusEnum, statusString). The following sections provide more details:

Sedona property data types to default Sedona proxy point types

Table 3-5 provides a cross listing of Sedona property data types, and the corresponding default Sedona (Niagara) proxy point types, and alternates.

Table 3-4 *Sedona property data types to default Proxy Point types, with alternate types*

Sedona data type	Default Proxy Point types	Alternately available point types
bool (boolean)	BooleanPoint, BooleanWritable	EnumPoint, EnumWritable (see "Booleans as Enums" on page 3-54)
float or double	NumericPoint, NumericWritable	—
int, long, byte, or short	EnumPoint, EnumWritable	NumericPoint, NumericWritable
Str (string), sys::Buf,	StringPoint, StringWritable	—

Sedona does not explicitly model "multistate" (enumerated) data. However, note that properties using integer type primitives (int, long, byte, short) default to Enum point types. Often, say for properties using "int" or "long" data type, you may wish to change this to NumericPoint or NumericWritable at proxy point add time.

Niagara writable null status and Sedona

Writable Sedona proxy points (BooleanWritable, NumericWritable, EnumWritable, StringWritable) have a "Fallback" property that is, by default, set to "null". When all other priority levels are cleared, a "null" status output results.

However, Niagara never issues any write to (or clears any value from) Sedona on a null transition—for example, whatever "null value" may exist in that Fallback property. Therefore, to remove any ambiguity you may want to clear (remove) the Fallback "null" setting, and enter (or verify) a specific fallback value to write from Niagara.

For example, consider a BooleanWritable proxy point for the "In" property of a Sedona "WriteBool" component. The BooleanWritable has a default Fallback property with the "null" status checked. If the proxy point is linked to a BooleanSchedule component that has its own "Default Output" of "- null", it is possible that all inputs to the proxy point will be nulled. However, nothing will be written to Sedona on the transition from schedule "On" to "null", such that the WriteBool will remain "true".

Also note that Sedona has its own "null" implementation for the primitive bool (boolean) type. For this, there is no direct translation from a Niagara "null" to a Sedona "null". However, if necessary the Sedona boolean null can be included in a Sedona proxy point. See "Booleans as Enums" for details.

Booleans as Enums

Although Sedona does not explicitly model "multistate" (enumerated) data, there is a unique exception of the boolean (bool) primitive type, with three possible states for boolean literals, as shown in Table 3-5.

Table 3-5 *Sedona boolean (bool) states*

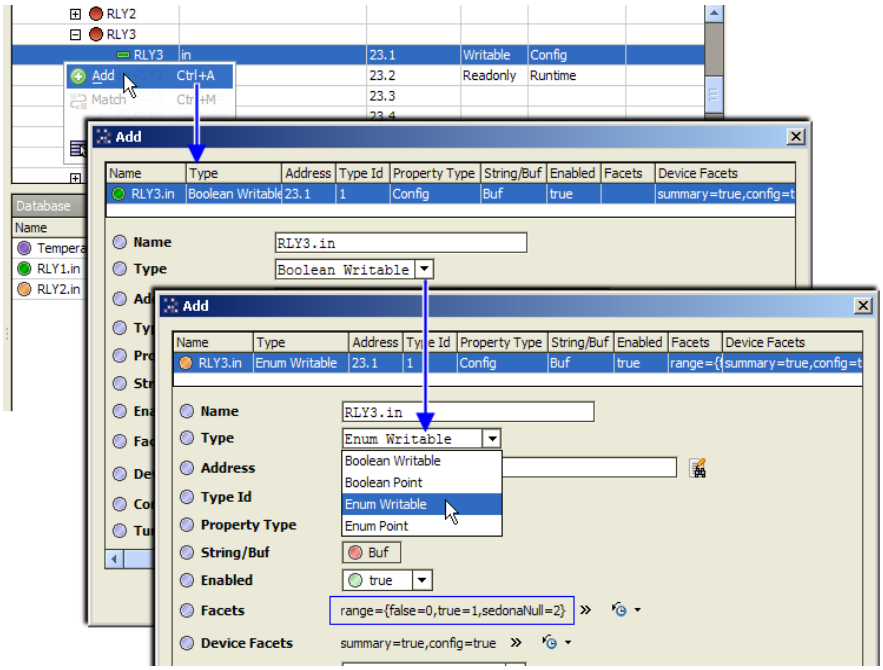
Value	Ordinal	String
false	0	"false"
true	1	"true"
null	2	"null"

In Sedona, null can indicate an invalid boolean value. If used in a boolean expression, null evaluates to true (it is represented as 2 in memory). Not all Sedona Framework apps may make use of boolean null.

Note: *The Niagara "null" status in writable Sedona proxy points does not translate into a "Sedona null", nor does a Niagara null status ever result in any write (i.e. "null value") to Sedona.*

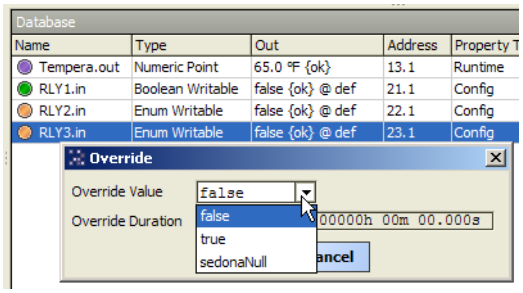
By default, a Sedona boolean property is proxied in Niagara as a BooleanPoint or (if writable) a BooleanWritable, with only two states. However, if you need to model a Sedona boolean property with all three states, you can select an EnumPoint (or if writable, an EnumWritable point) instead of a BooleanPoint or BooleanWritable, as shown being done in Figure 3-54.

Figure 3-54 *Re-selecting an Enum point type instead of Boolean for a Sedona "bool" property*



In this case, the Enum point's facets are automatically configured using the value and ordinals shown in Table 3-5, except null appears as "sedonaNull" (so as to distinguish this from when point *status* is "null"). If needed, you can edit these descriptors in point facets.

Figure 3-55 *Override action menu of EnumWritable that proxies a Sedona boolean (default descriptors)*



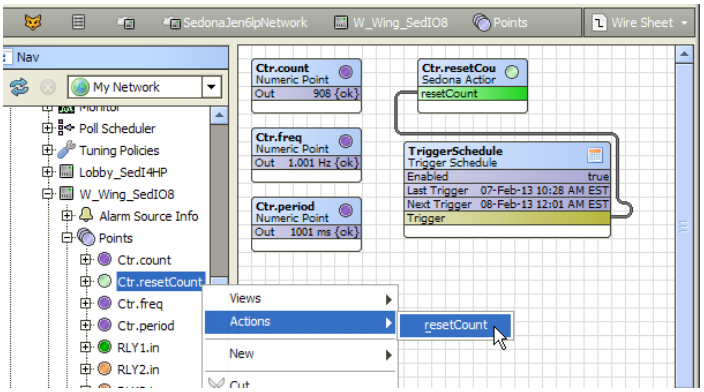
Note if an EnumWritable proxy point, these facet descriptors appear in the "override" action menu for the point, just like any other EnumWritable. See Figure 3-55.

About Sedona action points

Sedona action points are simple components, where each has a single action slot that represents a specific action on a Sedona component. You add and manage them from the Sedona Point Manager view of a Sedona Device's Points extension, along with Sedona proxy points.

In general, Sedona action point usage may be infrequent. One use case may be where a programmatic action is required from Niagara, such as the scheduled reset of an ongoing count (shown in Figure 3-56).

Figure 3-56 *Example Sedona action point linked into control logic*



As shown above, the Sedona action is available as a right-click action in Niagara. It is also an available link target for a programmatic invoke from control logic—in this case from a TriggerSchedule.

Chapter 4

Sedona Framework 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. Find documentation on a view by selecting **Help > On View** (F1) from the menu bar, or pressing F1 while in the view.

In this section, summary descriptions are given for views in the jen6lp, pansheet, and sedonanet modules, and a few details are given about views in the nsedona module. Summary descriptions typically include links to more detailed information.

Plugins in jen6lp module

The `jen6lp` module contains the following plugins (views):

- Sedona Jen6lp Device Manager

jen6lp-Sedona Jen6lp Device Manager

The **Sedona Jen6lp Device Manager** is the default view on a SedonaJen6lpNetwork component. It is used to manage SedonaJen6lpDevice children, where each represents a wireless Jennic-based device in the network where the JACE (with installed Sedona Jennic option card) acts as the "coordinator" node. This device manager offers online device discovery. For more details, see "Sedona Jen6lp Device Manager view" on page 3-32.

Plugins in nsedona module

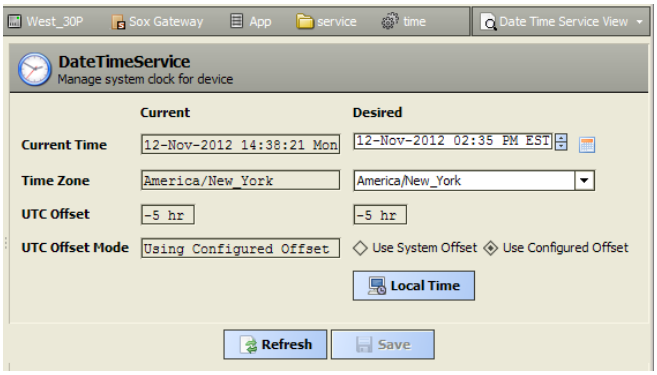
The `nsedona` module contains the following plugins (views), listed alphabetically:

- Date Time Service View
- Sedona Environment Manager
- Sedona Link Sheet
- Sedona Property Sheet
- Sedona Slot Sheet
- Sedona User Manager
- Sedona Wire Sheet

nsedona-Date Time Service View

The **Sedona Date Time Service** view the *default* view of the "time" service component (in the **service** folder), either a DateTimeServiceJenRtc or DateTimeServiceStd component.

Figure 4-1 *Date Time Service View*



This view provides a way to set the current time and date in the Sedona Framework node. Do this by typing and using the field controls for time and date (and) and time zone ( ), or by simply clicking the **`Local Time`** button to copy the Workbench PC's time and date to this node

Note: *After any* **`Save`** *in the Date Time Service View, also save the top-level "App" component.*

nsedona-Sedona Environment Manager

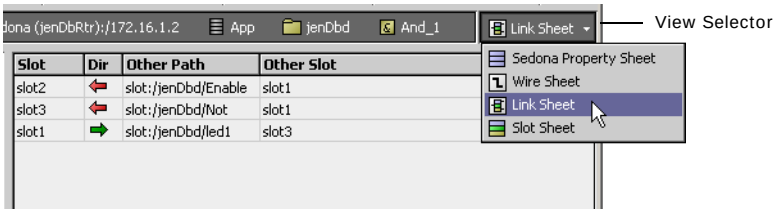
Starting in AX-3.7 with Sedona TXS-1.2, the **`Sedona Environment Manager`** view is an available platform view when platform connected to any remote AX-3.7 or later host. You use it to install and manage "Sedona environment files" on the remote host (typically a JACE). Included in this environment are Sedona kit manifest files, kit files, and platform database files.

For related procedures, see "JACE workflow" on page 1-2 and "Installing Sedona environment files in a JACE" on page 1-3. For an overview, see "Sedona environment management" on page B-1, with complete details on this view in "Sedona Environment Manager" on page B-3.

nsedona-Sedona Link Sheet

The **`Sedona Link Sheet`** view (link sheet) is an available view for any Sedona Framework component, providing details about any *link to* (inbound) or *link from* (outbound) that component. Figure 4-2 shows an example link sheet for a component.

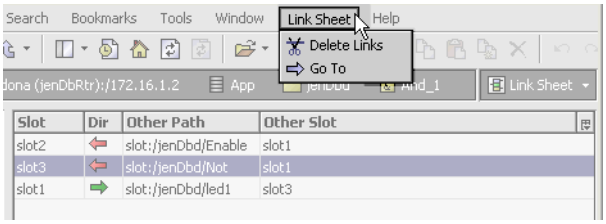
Figure 4-2 *Example Sedona Link Sheet view (And component)*



Each row represents a link, either an inbound link or an outbound link (seen in "Dir" column). Other columns provide information on the local slot ("Slot"), other component ("Other Path"), and the slot of that other component ("Other Slot").

When a link sheet view is active, the Workbench menu bar includes a Link Sheet menu (Figure 4-3).

Figure 4-3 *Link Sheet tool bar menu*



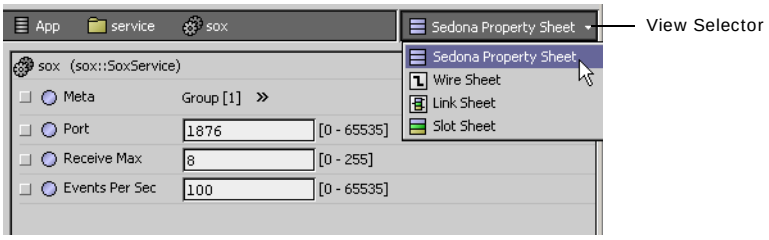
Tool bar menu items to **`Delete Links`** (one or more links selected) or **`Go To`** linked component (single link selected) are also available in the link sheet via a right-click menu.

nsedona-Sedona Property Sheet

The **Sedona Property Sheet** view (property sheet) is a standard view for any Sedona Framework component, providing access to the *properties* for that component. If a container type component, it also includes any child components as expandable [+] nodes—for example, the property sheet for a Folder component, will include a node for each component that it contains.

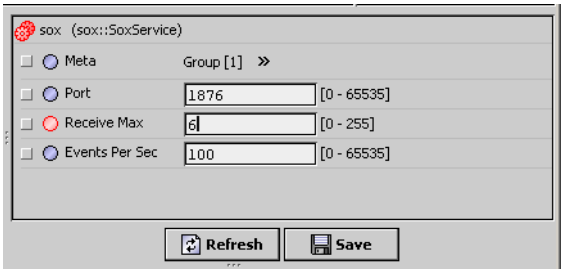
For many components, the property sheet view is the *default* view. The default view appears at the *top* of the "View Selector" for a component, as shown in Figure 4-4 (property sheet for SoxService).

Figure 4-4 *Example Sedona Property Sheet view (SoxService)*



In a property sheet, some properties may appear with a gray (vs. white) background—these are typically read-only properties—a common example is the property "Out". When you click on a *writable* property and enter a value, its appearance changes red to indicate a save is needed, and the **Save** button at the bottom of the view becomes available (Figure 4-5).

Figure 4-5 *Unsaved (ineffective) property sheet change indicated by red*



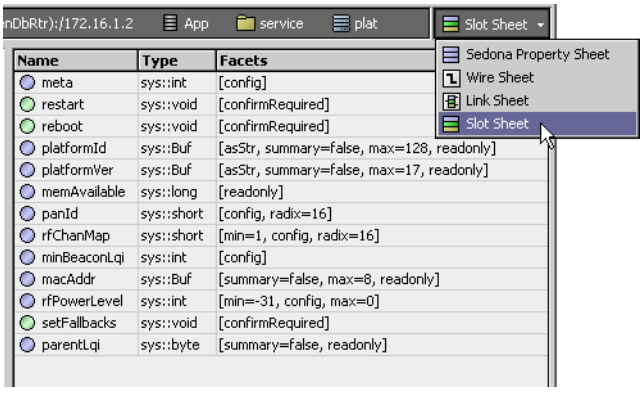
Click the **Save** button to write the change to the node's RAM, making it effective. However, note that such changes are not persisted in Flash memory of the device unless you save the App node.

For details on the "Meta" property in the Sedona property sheet of any component, see "nsedona-Sedona-Component (anyKit:anyComponent)" on page 5-3

nsedona-Sedona Slot Sheet

The **Sedona Slot Sheet** view (slot sheet) is a standard view for any Sedona Framework component, providing read-only data about both *properties* and *actions* for that component. Figure 4-6 shows an example slot sheet for a component.

Figure 4-6 *Example Sedona Slot Sheet view (jennic:JennicRouter component)*



Names of slots use a camelCase format, for example "minBeaconLqi". Whereas, those same slots appear listed on property sheet views by converting this format to include capital letters and spaces, for example "Min Beacon Lqi".

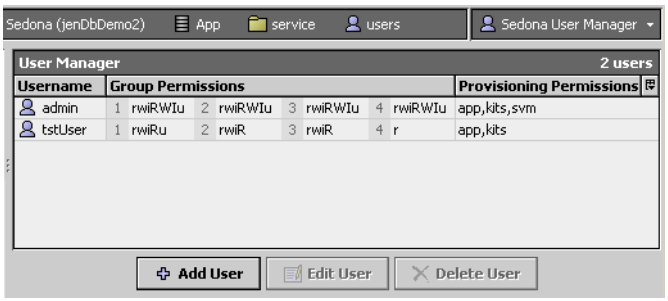
The "Type" column shows a property's primitive data type (where applicable), and the "Facets" column shows other data, such as if an action produces a confirmation dialog "[confirmRequired]".

nsedona-Sedona User Manager

The **Sedona User Manager** view is the *default* view of the "users" service component (in the **service** folder) of a Sedona Framework app (the sys:UserService component).

Note: *Starting in Sedona TXS-1.2, Sedona devices networked in a NiagaraAX station (SedonaNetwork or SedonaJen6lpNetwork) can optionally have Sedona users centrally managed at the station level. This is done using a* **Sedona User Manager** *service component (SedonaUserManagementService), along with a special* **Sedona Device User Manager** *view on the Sedona network component. For details, see* *"Sedona users and roles management" on page C-1.*

Figure 4-7 *Sedona User Manager view*



This view provides a way to view, add, edit, and delete child users in the app. Properties of each user determine their app login credentials, as well as their Sedona component permissions (across four groups), and sox provisioning permissions (across three different areas).

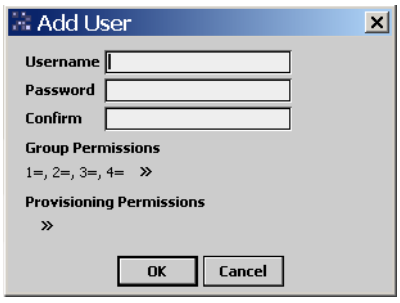
This view includes a User table and three User Manager buttons.

User table The table area lists each user on a separate row, summarizing in columns:

- **Username**
  Name of app user (7 characters maximum)
- **Group Permissions**
  Available permissions for that user among component groups 1-4, where:
  Operator level permissions show as "`r`" (read), "`w`" (write), "`i`" (invoke, action)
  Admin level permissions show as "`R`" (Read), "`W`" (Write), "`I`" (Invoke, action), "`u`" (user)
- **Provisioning Permissions**
  Available sox provisioning permissions for that user among three areas, where:
  Sedona app provisioning is "`app`", kit provisioning is "`kits`", and Sedona VM provisioning is "`svm`"

User Manager buttons The **Add User** button is always available. An Add User dialog results (Figure 4-8).

Figure 4-8 *Add User dialog*

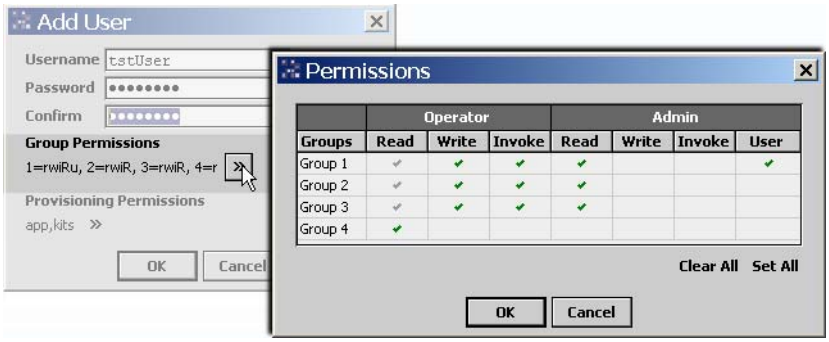


Configuration fields in Add User dialog are as follows:

- **Username**
  Unique name of user, 7 alphanumeric characters maximum, and is case sensitive. Must begin with letter and have no spaces or punctuation, apart from underscore (_).
- **Password**
  Login password for this user, case sensitive.

- **Confirm**
  Repeat of the same password for this user, case sensitive.
- **Group Permissions**
  Component permissions for the 4 different component groups can be enabled or disabled by clicking in the popup Permissions dialog, with two access levels: Operator and Admin (Figure 4-9).

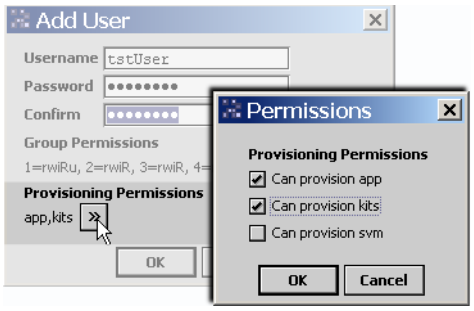
Figure 4-9 *Group Permissions dialog for a User*



If a component belongs to more than one group, the user has the highest of those different group permissions for that component.

- **Provisioning Permissions**
  Sox provisioning permissions for the 3 different areas of the platform can be enabled or disabled by clicking in the popup Permissions dialog (Figure 4-10).

Figure 4-10 *Provisioning Permissions dialog for a User*



The **Edit User** button is available when any one user (row) is selected. It produces an **Edit User** dialog that is similar to the **Add User** dialog (Figure 4-8), where changes can be made to any field *except* Username.

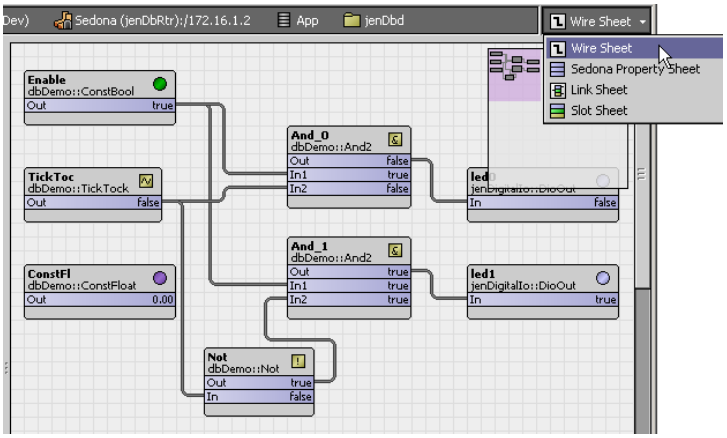
The **Delete User** button is available when any one user (row) is selected. It produces an **Delete User** confirmation dialog. If confirmed with **Yes**, that child User component is deleted.

nsedona-SedonaWireSheet

The **Sedona Wire Sheet** view (wire sheet) is a standard view for any Sedona Framework component. However, the main usage is with container-type components (notably Folder).

The wire sheet view is the *default* view for a Folder component, as shown in Figure 4-11.

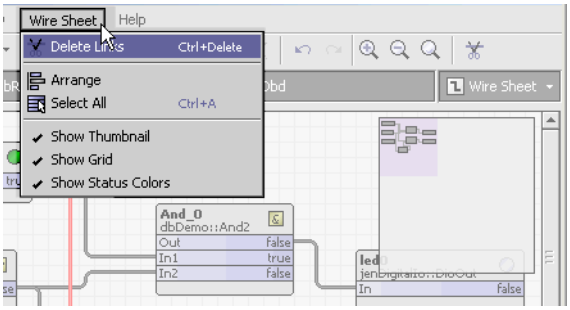
Figure 4-11 *Example Sedona Wire Sheet view (Folder)*



Wire sheet views are used to move components and link together, to graphically define control logic.

When a wire sheet view is active, the Workbench menu bar includes a Wire Sheet menu (Figure 4-12).

Figure 4-12 *Wire Sheet tool bar menu*



Tool bar menu items to **`Delete Links`** (one or more links selected), **`Arrange`** shapes, and **`Select All`** are also available in the wire sheet via a right-click menu. Enable/disable options for showing the navigation thumbnail, grid, and status colors are only in the tool bar menu.

Note: *To change the default wire sheet view settings, including the thumbnail, from the Workbench* **`Tools`** *menu, click the* **`Options`** *entry, then click* **`Wire Sheet`** *in the left-side options tree.*

Plugins in pansheet module

The `pansheet` module contains the following plugins (views):

- Pan Sheet

pansheet-Pan Sheet View

The **`Pan Sheet`** is an available diagnostic view on a SedonaJen6lpNetwork. It provides a graphical, hierarchical representation of the "JenNet" tree structure of the wireless network. The view also includes a "Pan Network" table showing "Pan Info" data collected throughout the network. Pan info data comes from SedonaJen6lpNeighborEntry slots in the SedonaJen6lpNetwork and all of its child SedonaJen6lpDevices. For details in this document, see "Pan Sheet view of SedonaJen6lpNetwork" on page 3-35. Also refer to the *Jennic Network Visualization (Pan Sheet) - Engineering Notes* document.

Plugins in sedonanet module

The `sedonanet` module contains the following plugins (views):

- Chopan Device Manager
- Chopan Point Manager
- Device Simulator View
- Manifest Manager
- Sedona Device Manager
- Sedona Device User Manager

- Sedona Point Manager
- Sedona Role Manager
- Sedona User Manager

sedonanet-Chopan Device Manager

The **Chopan Device Manager** is the default view of a **Virtual Chopan Network** under a SedonaJen6lpDevice's **Chopan Virtual** gateway. In this view, you can discover and create "Chopan devices", which result in *Sedona components* created in the remote Jennic-based device. Chopan devices (and Chopan points below them) act as the "Chopan client" interface in the device. For an overview in this document, see "About the Chopan Virtual gateway" on page 3-40. For complete details, refer to the *Sedona Framework Chopan Usage* document.

sedonanet-Chopan Point Manager

The **Chopan Point Manager** is the default view of a **Virtual Chopan Device** in a **Virtual Chopan Network** under a SedonaJen6lpDevice's **Chopan Virtual** gateway. In this view, you can discover and create "Chopan points", which result in *Sedona components* created in the remote Jennic-based device. Chopan points act as the "Chopan client" interface in the device.For an overview in this document, see "About the Chopan Virtual gateway" on page 3-40. For complete details, refer to the *Sedona Framework Chopan Usage* document, including the section "Chopan Point Manager notes".

sedonanet-Device Simulator View

The **Device Simulator View** is an available tool to run Sedona apps in a special "simulator" SVM (Sedona Virtual Machine) locally on your Workbench PC. This allows you diagnose problems with app logic, or to observe how a Sedona device and app can interact with a Niagara station.

Simulator SVMs vary according to specific Sedona hardware platforms, and must be obtained from the manufacturer (or vendor) of Sedona Framework devices. Such devices must implement Sedona Framework 1.2 or higher. For complete details, refer to the *Sedona Framework TXS Offline Engineering Guide*.

sedonanet-Manifest Manager

Use the **Manifest Manager** to manage Sedona kit *manifest* files, which are necessary for Sox access of Sedona Framework devices, both by a JACE station and by Workbench. The Manifest Manager is an available view on a Sedona Framework network component (SedonaNetwork or SedonaJen6lpNetwork) for managing manifests on a JACE. Sedona-enabled Workbench also has Sedona Manifest Manager (menu item **Tools > Sedona Manifest Manager**), for managing manifests on the Workbench host.

For a quick start procedure using the network-level **Manifest Manager**, see "Run the Manifest Manager to ensure kit manifests are loaded" on page 2-12. For complete Manifest Manager details, refer to the document *Sedona Framework Manifest Manager - Engineering Notes*.

sedonanet-Sedona Device Manager

The **Sedona Device Manager** is the default view on a SedonaNetwork component, as well as a SedonaDeviceFolder. It is used to manage SedonaDevice children, where each represents an Ethernet/IP equipped Sedona Framework device. This device manager does not offer online device discovery—instead you manually add devices using the **New** button and **New** dialogs. For more details, see "Sedona Device Manager view" on page 3-8.

sedonanet-Sedona Device User Manager

The **Sedona Device User Manager** is an available view on a SedonaNetwork or SedonaJen6lpNetwork. Use it to add Sedona users to the network's Sedona devices, and assign them user roles from the role database. For a summary, see "Sedona Device User Manager view" on page 3-11. For complete details, see the appendix "Sedona users and roles management", including subsections "Sedona user management overview" and "Centrally managing Sedona users".

sedonanet-Sedona Point Manager

The **Sedona Point Manager** is the default view on the SedonaPointDevExt (**Points**) extension of any Sedona device component (SedonaDevice, SedonaJen6lpDevice). In this view, you can discover, create, edit, and delete Sedona proxy points and action points. The Sedona Point Manager is also the default view for any SedonaPointFolders under the Points container. For more details, see "About the Sedona Point Manager" on page 3-47.

sedonanet-Sedona Role Manager

The **Sedona Role Manager** is the default view on the **Role Service** (SedonaRoleService) under the **Sedona User Manager** service (SedonaUserManagementService) in a station's Services container. (The other is the **User Service**, or SedonaUserService).

Use the **Sedona Role Manager** to manage a database of roles that can later be assigned to specific Sedona users when you assign them to devices. Each role is a unique ordered list (3-tuple) of role name, permissions, and provisioning rights. Deleting or modifying a role causes a synchronization task to be scheduled for all Sedona devices. For complete details, see the appendix "Sedona users and roles management", including subsections "Sedona user management overview" and "Centrally managing Sedona users".

sedonanet-Sedona User Manager

The **Sedona User Manager** is the default view on the **User Service** (SedonaUserService) under the **Sedona User Manager** service (SedonaUserManagementService) in a station's Services container. (The other is the **Role Service**, or SedonaRoleService).

Use the **Sedona User Manager** to manage a set of Sedona users that can later be assigned to specific Sedona devices. Deleting or modifying a user causes a synchronization task to be scheduled for all Sedona devices. For complete details, see the appendix "Sedona users and roles management", including subsections "Sedona user management overview" and "Centrally managing Sedona users".

Chapter 5

Sedona Framework Component Guides

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

- jen6lp
- nsedona
- sedonanet

Components in jen6lp module

The `jen6lp` module contains the following components, listed alphabetically:

- Ipv4Mac154Table
- PanInfoPoller
- SedonaJen6lpDevice
- SedonaJen6lpDeviceFolder
- SedonaJen6lpNeighborEntry
- SedonaJen6lpNetwork
- SedonaJen6lpPingMonitor

jen6lp-Ipv4Mac154Table

Ipv4Mac154Table (**`Address Map`**) is a child container under a SedonaJen6lpNetwork. Its configuration property "Ipv4 Address Base" defines the private IPv4 subnet to which Jennic-based nodes are remapped by the JACE coordinator. It also contains dynamically-created "entry*N*" container slots that define each node's mapping, including specific IPv4 address, MAC 15.4 address, and JenNet "join" status. An available "Clear Table" action clears out all "entry*N*" children—used only if reconfiguring the network.

For more details, see "SedonaJen6lpNetwork coordinator properties" on page 3-29.

jen6lp-PanInfoPoller

The PanInfoPoller (**`PanInfo Poller`**) is a frozen container slot of a SedonaJen6lpNetwork. It specifies how "PAN info" (JenNet related) data is received, either by ongoing polling or "on demand". The default is polling disabled (property Enabled = false), with property "Use Chopan" as true (vs. Sox). Chopan usage requires that Jennic-based devices have a Sedona Framework app with both the PanInfoService and the ChopanService installed, as well as the PanInfoServlet added under the ChopanService.

PAN info from child nodes is modeled dynamically under the SedonaJen6lpNetwork's frozen container component **`Child Pan Info`**, in SedonaJen6lpNeighborEntry components. The available Pan Sheet view of the SedonaJen6lpNetwork provides a graphical representation of such paninfo.

For more details, see "SedonaJen6lpNetwork Pan Info properties" on page 3-30, "Pan Sheet view of SedonaJen6lpNetwork" on page 3-35, and "SedonaJen6lpDevice properties" on page 3-36.

jen6lp-SedonaJen6lpDevice

SedonaJen6lpDevice represents a wireless Jennic-based device, and is the only valid type of device object in a SedonaJen6lpNetwork. As a subclass of the SedonaDevice, it has the same common device properties and Points extension, plus additional properties and container slots for support of wireless Jennic communications as well as Chopan. For more details, see "About the SedonaJen6lpDevice component" on page 3-36.

jen6lp-SedonaJen6lpDeviceFolder

SedonaJen6lpDeviceFolder is the implementation of a device folder under a SedonaJen6lpNetwork. You can use these folders to organize SedonaJen6lpDevice components in the network.

Typically, you add such folders using the **New Folder** button in the Sedona Jen6lp Device Manager view of the SedonaJen6lpNetwork. Each SedonaJen6lpDeviceFolder has its own Sedona Jen6lp Device Manager view. The SedonaJen6lpDeviceFolder is also available in the `jen6lp` palette.

jen6lp-SedonaJen6lpNeighborEntry

A SedonaJen6lpNeighborEntry (MAC*802.15.4MacAddress*) is a dynamic child container that stores "paninfo" data about a nearby (neighbor) Jennic node. Its read-only properties report if the device is a sleeping end device, its MAC 15.4 address, and several numerical statistics including a link quality index, and numbers of packets lost, sent, and received. These components can be either "child" or "parent" in nature, depending on the context of component hierarchy.

- In the SedonaJen6lpNetwork, its frozen **Child Pan Info** slot holds a number of child SedonaJen6lpNeighborEntries, one for each direct child node. Because the network represents the (top level) coordinator, it has no parent entries.
- Each SedonaJen6lpDevice has a **Parent Pan Info** slot, of type SedonaJen6lpNeighborEntry, with data about its one (and only) Jennic parent node. Depending on the device's location and the network's "depth", this parent node may either be the coordinator (JACE option card), or another Jennic-based device acting as a "router" node.
  Each SedonaJen6lpDevice also has a **Child Pan Info** container slot, which may (or may not) have SedonaJen6lpNeighborEntry children.

The entire collection of data from all SedonaJen6lpNeighborEntries, network-wide, a is reflected *graphically* in the Pan Sheet view of the SedonaJen6lpNetwork. This can be useful for diagnostic purposes. For related details in this document, see the following sections:

- "SedonaJen6lpNetwork Pan Info properties" on page 3-30.
- "SedonaJen6lpNetwork actions" on page 3-32.
- "Pan Sheet view of SedonaJen6lpNetwork" on page 3-35.
- "SedonaJen6lpDevice properties" on page 3-36.
- "SedonaJen6lpDevice actions" on page 3-39.

Find complete details about using the **Pan Sheet** in the *Jennic Network Visualization (Pan Sheet) - Engineering Notes* document.

jen6lp-SedonaJen6lpNetwork

SedonaJen6lpNetwork is a subclass of the base SedonaNetwork, used for managing a network of wireless (802.15.4) Jennic-based devices in a station running in a JACE with an installed Sedona Jennic option card. Devices are represented by child SedonaJen6lpDevice components, mapped from their native IPv6 (6LoWPAN) addresses to a private IPv4 address range defined in the network.

The SedonaJen6lpNetwork extends the base SedonaNetwork, providing additional properties, child containers, and views that help configure and troubleshoot the Jennic wireless network.

As with other NiagaraAX driver networks, the SedonaJen6lpNetwork should reside under the station's Drivers container. The default view of the network is the **Sedona Jen6lp Device Manager**.

For more details, see "About the SedonaJen6lpNetwork component" on page 3-23 and "Sedona Jen6lp Device Manager view" on page 3-32.

jen6lp-SedonaJen6lpPingMonitor

The SedonaJen6lpPingMonitor (**Monitor**) is a frozen container slot of a SedonaJen6lpNetwork. It periodically checks the Jennic node status of devices in the local coordinator to determine the corresponding child Jen6lpDevices' health. PingMonitor provides built-in support to generate alarms when these pingables are down. See "About Monitor" in the *Drivers Guide* for general monitor details.

Components in nsedona module

The `nsedona` module contains the following components, listed alphabetically:

- DaspTunnel (SoxTunnel)
- SedonaComponent (*anyKit:anyComponent*)
- SoxSession

nsedona-DaspTunnel

DaspTunnel (**SoxTunnel**) is the station service that allows Sox tunneling through the running (JACE) station to a Sedona Framework device. This allows Niagara Workbench to access Sox Tools and app configuration for Sedona Framework devices that are modeled in the station, in either (or both) a SedonaNetwork or SedonaJen6lpNetwork. The best practice is to copy the SoxTunnel into the station's Services folder.

Note: *To use this feature, the NiagaraAX station host as well as any Niagara Workbench client must have a license with the* `tunneling` *feature, with its attribute* `sox="true"`.

The SoxTunnel service contains several properties, often which can be left at defaults. For complete details, refer to the document *Sedona Framework Sox Tunneling - Engineering Notes*.

nsedona-SedonaComponent *(anyKit:anyComponent)*

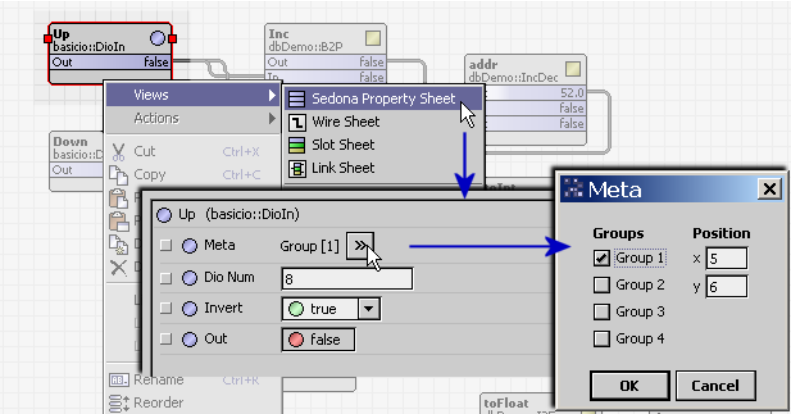
Represents any Sedona Framework component. All Sedona Framework components provide the identical "Guide on Target" Help ord in Workbench, instead of unique ords by *kitName-componentName*. Therefore this section describes the single common property implemented by all Sedona Framework components:Meta.

Note: *Refer to the vendor's documentation for any Sedona Framework device for specific details on Sedona components used in custom kits, and also the Sedona Framework Developer's website for details on components in open source kits. At the time of this document, the URL for this API documentation is:*

http://www.sedonadev.org/doc/api.html

Meta The Meta property is available in property sheet of any Sedona Framework component, with Figure 5-1 showing an example.

Figure 5-1 *Meta property example in Sedona component*



Meta stores two different pieces of information:

- **Groups**
  Four component groups are available, numbered 1 through 4. Each component *must* belong to at least one group, and can belong to all four if desired. Membership in a group determines what permissions an app *user* has for that component, according to that user's permissions map.
  For related details, see "nsedona-Sedona User Manager" on page 4-4.
- **Position**
  The x and y coordinates of the component's upper-left corner's glyph (shape) in the wire sheet view of its parent container. Integer values are used, which correspond to grid increments on the wire sheet, starting with 0, 0 at the wire sheet's top-left corner.
  A component's position values automatically track any "drag" movement of its shape on a wire sheet, and thus typically are not directly edited in the Meta dialog (Figure 5-1).
  Storing the position of the components in an app maintains wire sheet layouts, which is essential when working with collections of components and the links between them.

nsedona-SoxSession

`Sedona` (*AppName*) represents a direct Sox connection from either Niagara Workbench or Sedona Workbench to a running Sedona Framework device.

If the session is tunneled through a Niagara station with the SoxTunnel service, it appears as **`Sedona`** (*AppName*):/*IpAddress* instead, and is under that station's host address space.

In either case, the default Nav Container View contains nodes for **`Sedona Tools`** and the **`App`** running in the device. For details on the Sedona Tools, refer to the *Sedona Framework TXS Sedona Tools Guide*.

Note: *Workbench acts as the "Sox client" with any direct Sox or tunneled Sox connection to a Sedona device. This means your Workbench requires, at a minimum, the kit manifest files associated with each device. If provisioning the device, the Sedona environment of your Workbench also requires all kit files and the platform archive file associated with that device. Note this differs from "Sox Gateway" access to a Sedona device, where the Sox client resides on the JACE/station.*

Components in sedonanet module

The `sedonanet` module contains the following components, listed alphabetically:

- ChopanServer
- ChopanVirtualGateway
- RefreshDeviceInfoTask
- SedonaActionPoint
- SedonaDevice
- SedonaDeviceFolder
- SedonaDeviceInfoExt
- SedonaNetwork
- SedonaPollScheduler
- SedonaPointDeviceExt
- SedonaPointFolder
- SedonaProxyExt
- SedonaRole
- SedonaRoleService
- SedonaToolsContainer
- SedonaTuningPolicy
- SedonaTuningPolicyMap
- SedonaUser
- SedonaUsersExt
- SedonaUserManagementService
- SedonaUserService
- SoxDeviceGateway
- UserSyncTask
- VirtualChopanBooleanPoint
- VirtualChopanDevice
- VirtualChopanFloatPoint
- VirtualChopanNetwork

sedonanet-ChopanServer

ChopanServer is a container slot under a SedonaJen6lpNetwork for configuration of the JACE station's CHoPAN server capability. Its properties specify the UDP port monitored, and whether the server is enabled or disabled. Debug output to the station's Application Director can also be enabled.

If configured for Chopan in their Sedona Framework app, this allows Jennic-based devices in the SedonaJen6lpNetwork to access station data via Chopan client requests. For more details, see "SedonaJen6lpNetwork Chopan Server" on page 3-28. For complete Chopan details, refer to the *Sedona Framework Chopan Usage* document.

sedonanet-ChopanVirtualGateway

The ChopanVirtualGateway is the SedonaJen6lNetwork implementation of the Baja Virtual Gateway. A virtual gateway is a component that resides under the station's component space (`Config`), and acts as a gateway to the station's "virtual component space."

Each SedonaJen6lpDevice has its own ChopanVirtualGateway, at the same level as its device extensions (Points, in this case). Using the gateway, you can discover nodes in the network that have CHoPAN servers in them—including the JACE. You can add Sedona components to the node's app that represent these CHoPAN servers, and then add components inside the servers that represent points within them. These are actual components inside the app of the physical device your SedonaDevice is mapping. However, they are presented in the form of Niagara virtual points, because the device and point addition is not taking place in the station, but on the device itself.

For details in this document, see "About the Chopan Virtual gateway" on page 3-40. For complete Chopan details, refer to the *Sedona Framework Chopan Usage* document.

sedonanet-RefreshDeviceInfoTask

RefreshDeviceInfoTask (**`Refresh Task`**) is a child container of the SedonaDeviceInfoExt (Device Info) of each Sedona device component. It contains properties Status and Fault Cause, and has actions Schedule, Execute, and Cancel. For related details, see "Sedona Device Info Ext" on page 3-14.

sedonanet-SedonaActionPoint

SedonaActionPoint is Niagara representation for a particular action slot of a Sedona component, as added under the **Points** extension of a SedonaDevice or SedonaJen6lpDevice using its **Sedona Point Manager** view. The SedonaActionPoint provides a single right-click action, which proxies the action available on the component in the device's Sedona Framework app.

Both Sedona proxy points and Sedona action points are visible in the Sedona Point Manager view of a Sedona Framework device. For more details, see "About Sedona action points" on page 3-55.

sedonanet-SedonaDevice

SedonaDevice represents an Ethernet/IP (or 802.11 WiFi) capable device, and is the only valid device object type in a SedonaNetwork. As the base class Sedona Framework device component, it has common device properties and a Points extension. See "About the SedonaDevice component" on page 3-12.

sedonanet-SedonaDeviceFolder

SedonaDeviceFolder is the implementation of a device folder under a SedonaNetwork. You can use these folders to organize SedonaDevice components in the network.

Typically, you add such folders using the **New Folder** button in the Sedona Device Manager view of the SedonaNetwork. Each SedonaDeviceFolder has its own Sedona Device Manager view. The SedonaDeviceFolder is also available in the `sedonanet` palette.

sedonanet-SedonaDeviceInfoExt

SedonaDeviceInfoExt (**Device Info**) is the frozen device extension under each Sedona device that contains read-only properties about its Sedona platform type. Values are read from properties of components in the device's Sedona app. For more details, see "Sedona Device Info Ext" on page 3-14.

sedonanet-SedonaNetwork

SedonaNetwork is the base class Sedona Framework network, used for managing a network of Sedona Framework devices equipped for Ethernet/IP. If properly licensed, the NiagaraAX station can run in a JACE or a Supervisor station. Devices are represented by child SedonaDevice components, the only valid device type for this network type.

As with other NiagaraAX driver networks, the SedonaNetwork should reside under the station's Drivers container. The default view of the SedonaNetwork is the **Sedona Device Manager**, where you manually add devices, specifying each device's unique IPv4 address.

For more details, see "About the SedonaNetwork component" on page 3-6 and "Sedona Device Manager view" on page 3-8.

sedonanet-SedonaPollScheduler

The SedonaPollScheduler is the Sedona implementation of a poll scheduler in a SedonaNetwork (or SedonaJen6lpNetwork). Operation is similar to most other polling networks.

sedonanet-SedonaPointDeviceExt

SedonaPointDeviceExt is the Sedona implementation of PointDeviceExt. Its primary view is the **Sedona Point Manager**. Currently it is the only device extension for any Sedona Framework device (SedonaDevice, SedonaJen6lpDevice). It serves as the container for all Sedona proxy points for a specific device. For more details, see "About the Sedona Point Manager" on page 3-47.

sedonanet-SedonaPointFolder

SedonaPointFolder is the Sedona implementation of a folder under a Sedona Framework device's **Points** container (SedonaPointDeviceExt). You add such folders using the **New Folder** button in the **Sedona Point Manager** view of the Points component. Each SedonaPointFolder has its own Sedona Point Manager view. The SedonaPointFolder is also available in the `sedonanet` palette.

sedonanet-SedonaProxyExt

SedonaProxyExt is the proxy extension for any type of Sedona proxy point, as added under the **Points** extension of a SedonaDevice or SedonaJen6lpDevice using its **Sedona Point Manager** view. The SedonaProxyExt contains common ProxyExt properties, as well as a few unique to Sedona proxy points. For more details, see "About Sedona proxy points" on page 3-52.

sedonanet-SedonaRole

Sedona Role is a child of the SedonaRoleService (**Role Service**), and represents a set of permission bits and provisioning bits that can be mapped to Sedona Users and assigned to Sedona devices. You create and manage Sedona Roles from the **Sedona Role Manager** (default) view of the SedonaRoleService (**Role Service**).

For related details, see "Sedona user management overview" on page C-2 and "Centrally managing Sedona users" in "Sedona users and roles management".

sedonanet-SedonaRoleService

The SedonaRoleService (**Role Service**) is a child of the SedonaUserManagementService (**SedonaUserManager**), and manages a set of Sedona Roles for the station. It has no frozen properties, but its default view is the **Sedona Role Manager**, used to add or edit Sedona Roles.

For related details, see "Sedona user management overview" on page C-2 and "Centrally managing Sedona users" in "Sedona users and roles management".

sedonanet-SedonaToolsContainer

The **Sedona Tools** container under each networked Sedona device provides access to the Sedona tools (**Kit Manager**, **Application Manager**, and **Backup/Restore Tool**) for provisioning that device, using the Sedona environment files installed on the Niagara host (typically JACE) running the station. The default view on this container is the **Sedona Tools** view, which lists the schema (collection of kits) installed in the device.

Note: *Before using these tools, installation of "Sedona environment files" are required on the Niagara host (typically a JACE) running this station, using a Niagara platform connection. For details, see "Sedona environment management" on page B-1.*

For information on the various Sedona Tools, refer to the *Sedona Framework TXS Sedona Tools Guide*.

sedonanet-SedonaTuningPolicy

A tuning policy for a SedonaNetwork (or SedonaJen6lpNetwork), with standard NiagaraAX tuning policy properties. For more details, see"SedonaNetwork tuning policy notes" on page 3-7 and "SedonaJen6lpNetwork tuning policy notes" on page 3-27.

sedonanet-SedonaTuningPolicyMap

SedonaTuningPolicyMap (**Tuning Policies**) is a SedonaNetwork (or SedonaJen6lpNetwork) container slot for one or more SedonaTuningPolicy(ies).

sedonanet-SedonaUser

Sedona User is a child of the SedonaUserService (**User Service**), and represents a potential user for one or more Sedona devices networked in the station. You create and manage Sedona Users from the **Sedona User Manager** (default) view of the SedonaUserService (**User Service**).

For related details, see "Sedona user management overview" on page C-2 and "Centrally managing Sedona users" in "Sedona users and roles management".

sedonanet-SedonaUsersExt

SedonaUsersExt (**Sedona Users**) is an extension of a Sedona device (SedonaDevice or SedonaJen6lpDevice), present whenever the station's Services contains a SedonaUserManagement-Service component. This component stores (internally) the mapping of users and roles for the device, and manages synchronization of those mappings down to the Sedona Framework device.

For related details, see "User Synchronization" on page C-13 in "Sedona users and roles management".

sedonanet-SedonaUserManagementService

SedonaUserManagementService (**SedonaUserManager**) is the container for Sedona user and role services for Sedona devices networked in a station. Available in the `sedonanet` palette, you copy it in a station's **Services** container for centralized management of Sedona Framework device users.

For related details, see "Sedona users and roles management" on page C-1.

sedonanet-SedonaUserService

SedonaUserService (**User Service**) is a child of the SedonaUserManagementService (**SedonaUserManager**), and manages a set of Sedona Users for the station. It has no frozen properties, but its default view is the **Sedona User Manager**, used to add or edit Sedona Users.

For related details, see "Sedona user management overview" on page C-2 and "Centrally managing Sedona users" in "Sedona users and roles management".

sedonanet-SoxDeviceGateway

The SoxDeviceGateway (**Sox Gateway**) appears under each networked Sedona device in a SedonaNetwork or SedonaJen6lpNetwork, and provides a "gateway" into that device's Sedona app. For details, see "Sox Gateway" on page 3-16.

sedonanet-UserSyncTask

UserSyncTask (**`Sync Task`**) is a child container of the SedonaUsersExt (Sedona Users) of any Sedona device component. This component works to synchronize changes in Sedona users and roles mapped to this device, from the Niagara definitions to the Sedona users in the device's app. It contains properties Status and Fault Cause, and has actions Schedule, Execute, and Cancel. For related details, see "User Synchronization" on page C-13 in "Sedona users and roles management".

sedonanet-VirtualChopanDevice

VirtualChopanDevice represents a "ChopanDevice" component in the Sedona Framework app of a remote Jennic-based device. In the JACE station, it is a child of the VirtualChopanNetwork under the ChopanVirtualGateway of a SedonaJen6lpDevice. It can contain child virtual "Chopan points" (Boolean or float types), where each represents a data item served in Chopan from another Jennic-based device (or the JACE station). The default view of a VirtualChopanDevice is the Chopan Point Manager, where you can discover and add virtual Chopan points. For an overview, see "About the Chopan Virtual gateway" on page 3-40. For more details, refer to the *Sedona Framework Chopan Usage* document.

sedonanet-VirtualChopanBooleanPoint

VirtualChopanBooleanPoint is one of two types of "Chopan points" (the other type is VirtualChopanFloatPoint). It represents a Boolean type (Chopan client) component in the Sedona Framework app of a remote Jennic-based device. In the JACE station, it is a child of a VirtualChopanDevice in a ChopanVirtualNetwork within the ChopanVirtualGateway of a SedonaJen6lpDevice. The source (Chopan server) device is another Jennic-based device, or if enabled, the JACE station via the Chopan server slot of the SedonaJen6lpNetwork. For an overview, see "About the Chopan Virtual gateway" on page 3-40. For more details, refer to the *Sedona Framework Chopan Usage* document.

sedonanet-VirtualChopanFloatPoint

VirtualChopanFloatPoint is one of two types of "Chopan points" (the other type is VirtualChopanBooleanPoint). It represents a float type (Chopan client) component in the Sedona Framework app of a remote Jennic-based device. In the JACE station, it is a child of a VirtualChopanDevice in a ChopanVirtualNetwork within the ChopanVirtualGateway of a SedonaJen6lpDevice. The source (Chopan server) device is another Jennic-based device, or if enabled, the JACE station (via the Chopan server slot of the SedonaJen6lpNetwork). For an overview, see "About the Chopan Virtual gateway" on page 3-40. For more details, refer to the *Sedona Framework Chopan Usage* document.

sedonanet-VirtualChopanNetwork

VirtualChopanNetwork is a "virtual component" child of the ChopanVirtualGateway in a SedonaJen6lpDevice. It represents the "ChopanNetwork" component in the Sedona Framework app of a remote Jennic-based device. In the JACE station, it can contain child VirtualChopanDevice components, each representing a Jennic-based device with Chopan installed, as well as one device representing the JACE. The default view of the VirtualChopanNetwork is the Chopan Device Manager, where you can discover and add these VirtualChopanDevice components. For an overview, see "About the Chopan Virtual gateway" on page 3-40. For more details, refer to the *Sedona Framework Chopan Usage* document.

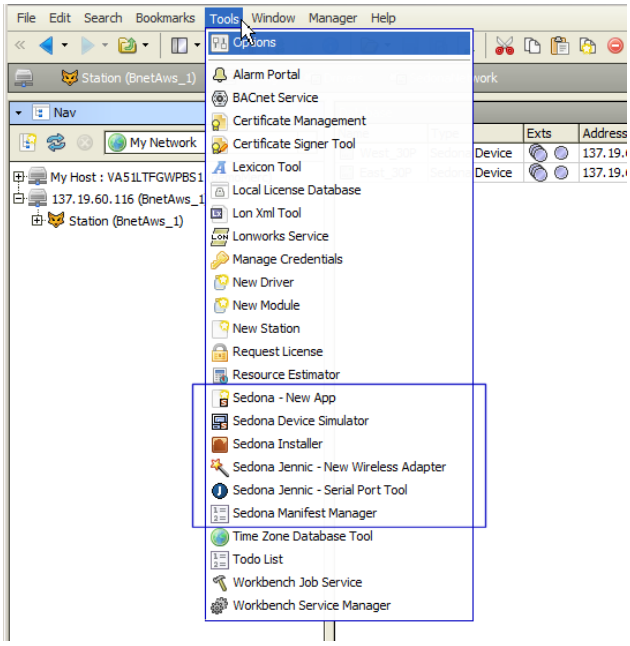
Appendix A

Sedona Framework Tools for Workbench

If Workbench has been enabled for Sedona Framework TXS-1.2 (using the Workbench tool **`Sedona Installer`**), several Sedona Framework-related "tools" become available in the Workbench **`Tools`** menu. This section provides overview information on these tools, along with links to any other documents with more details.

Note: *The Sedona Installer is already included in any Workbench distribution, and is how you first enable Niagara Workbench for Sedona Framework TXS-1.2. Afterwards, you can also use it to "upgrade" the Sedona Framework-specific NiagaraAX modules and files for your Workbench (by installing a Bundle), and/or also to import any "Sedona environment files" for your Workbench.*

Figure A-1 *Sedona Framework-related tools in Tools menu of Niagara Workbench*

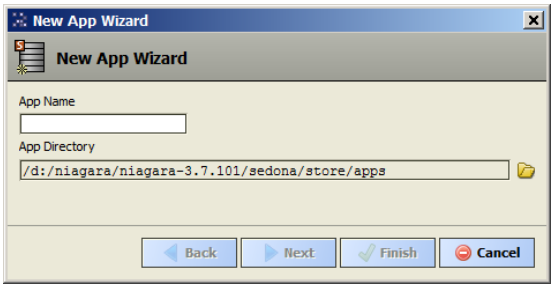


The following Sedona Framework-related Workbench tools are included:

Sedona - New App

The **`Sedona - New App`** tool is a wizard to start a new Sedona app in offline mode.

Figure A-2 *Sedona - New App (wizard)*



Steps in the wizard have you specify:

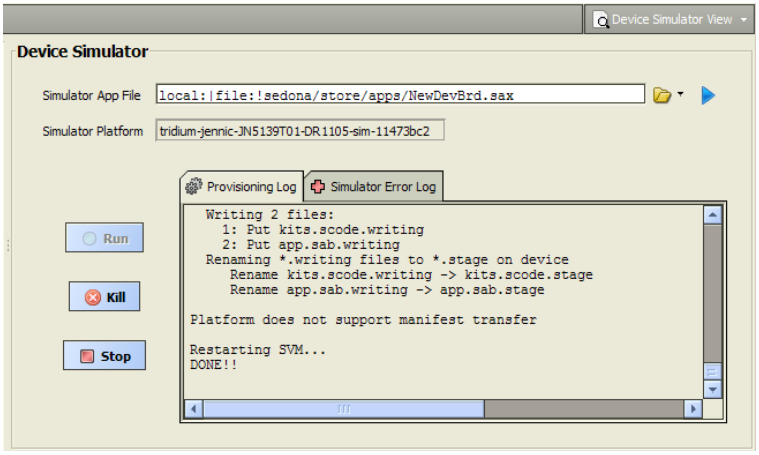
- The app name and location on your Workbench PC to create the Sedona app (.sax file).
- Password for the Admin user in the app, and the Sox port used to connect to the device.
- Target Sedona platform type (from the platforms available in your Workbench PC's Sedona environment), and the device name to use in the app.

You can choose for the wizard to end by opening the Sedona app in the Workbench view to continue offline engineering. For more details, refer to the *Sedona Framework TXS Offline Engineering Guide*.

Sedona Device Simulator

The **`Sedona Device Simulator`** tool lets you run a Sedona app on your Workbench PC as a "simulated device", using the Sedona device vendor-supplied simulator SVM (Sedona virtual machine).

Figure A-3 *Sedona Device Simulator*



This allows you diagnose problems with app logic, or to observe how a Sedona device and app can interact with a Niagara station.

Simulator SVMs vary according to specific Sedona hardware platforms, and must be obtained from the manufacturer (or vendor) of Sedona Framework devices. Such devices must implement Sedona 1.2 or higher. For complete details, refer to the document *Sedona Framework TXS Offline Engineering Guide*

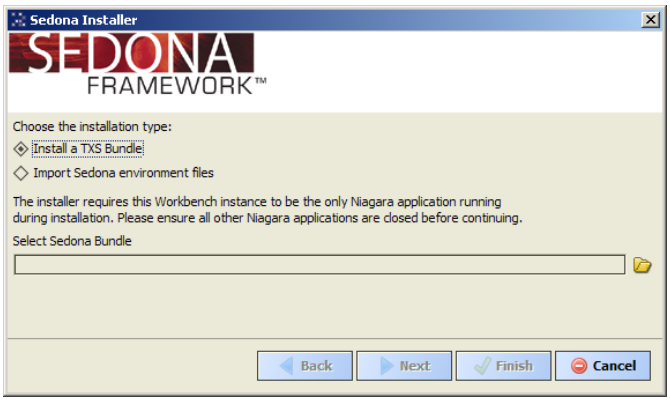
Sedona Installer

The **`Sedona Installer`** is the tool you use to upgrade a standard installation of Niagara Workbench to be "Sedona-enabled" Niagara Workbench, using a "Sedona bundle" image zip file. After this initial installation, you continue to use the Sedona Installer for applying Sedona bundle updates to your Workbench platform.

Note: *Sedona includes licensed features for Workbench as well as JACE hosts with stations that use Sedona Framework network drivers.*

Starting in AX-3.7 and also Sedona Framework Workbench 1.2, the Sedona Installer also provides an option for installing "Sedona environment files". This allows you to install device vendor-supplied Sedona kit manifest files, Sedona kit files, and Sedona platform archive files in your "local" (Workbench) Sedona environment.

Figure A-4 *Sedona Installer is used to install "Sedona bundle" image distributions*

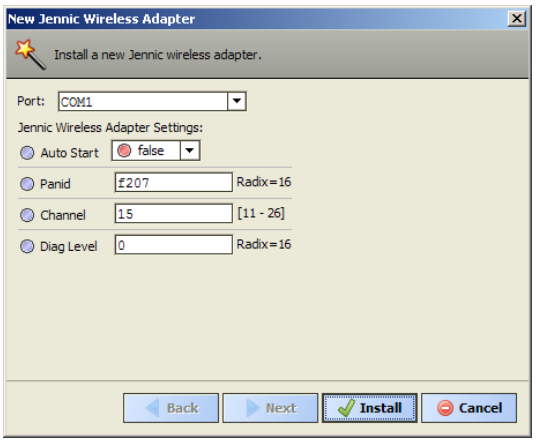


For complete details, including using both options as well as Workbench licensing information, see the *NiagaraAX Sedona Installer Guide*.

Sedona Jennic - New Wireless Adapter

The **`Sedona Jennic - New Wireless Adapter`** tool is a wizard you can use to identify and configure a new wireless Sedona Jennic USB coordinator (USB stick) for use with your Sedona Framework TXS enabled Niagara Workbench PC.

Figure A-5 *Sedona Jennic- New Wireless Adapter wizard*



Note: *Prior to using this tool, you need to download and install the latest FTDI (Future Technologies Devices International) "VCP driver" for your system. After you use the Windows Device Manager to determine the COM port number assigned to this "virtual com port", use this tool to configure the USB coordinator.*

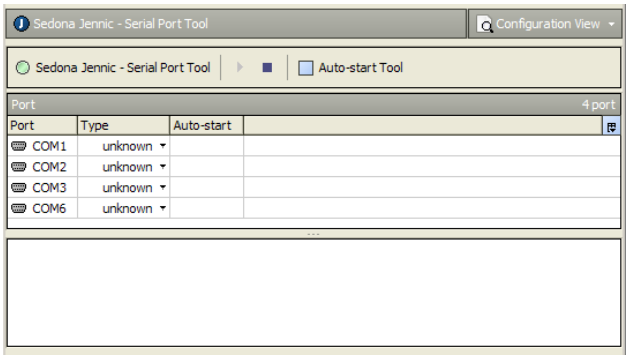
For more details, refer to the document *Jennic Serial Tools Guide*.

Sedona Jennic - Serial Port Tool

The **`Sedona Jennic - Serial Port Tool`** allows you to reconfigure a wireless Sedona Jennic USB coordinator (USB stick), if available, for direct Sox connection from Workbench to a single Jennic-based device. For example, to change its Panid, channel, or debug level.

This tool also provides a "flasher" utility to download firmware into a serially attached Jennic-based device (special cable typically required), or to update the firmware in a Sedona Jennic USB coordinator.

Figure A-6 *Sedona Jennic - Serial Port Tool*

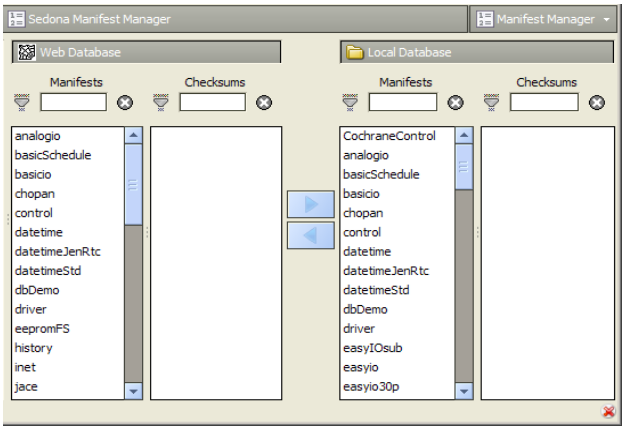


For more details, refer to the document *Jennic Serial Tools Guide*.

Sedona Manifest Manager

The **`Sedona Manifest Manager`** is a tool you can use to manage Sedona kit manifest files on your Sedona-enabled Niagara Workbench platform.

Figure A-7 *Sedona Manifest Manager*



For complete details, see the *Sedona Framework Manifest Manager - Engineering Notes* document.

Appendix B

Sedona environment management

This section explains managing the "Sedona environment" on any remote AX-3.7 or later host with Sedona Framework TXS-1.2 (typically a JACE). Such configuration is required to support the "provisioning through the station" feature available for Sedona devices networked under a JACE station, as well as "Sox Gateway" access to Sedona apps in networked devices.

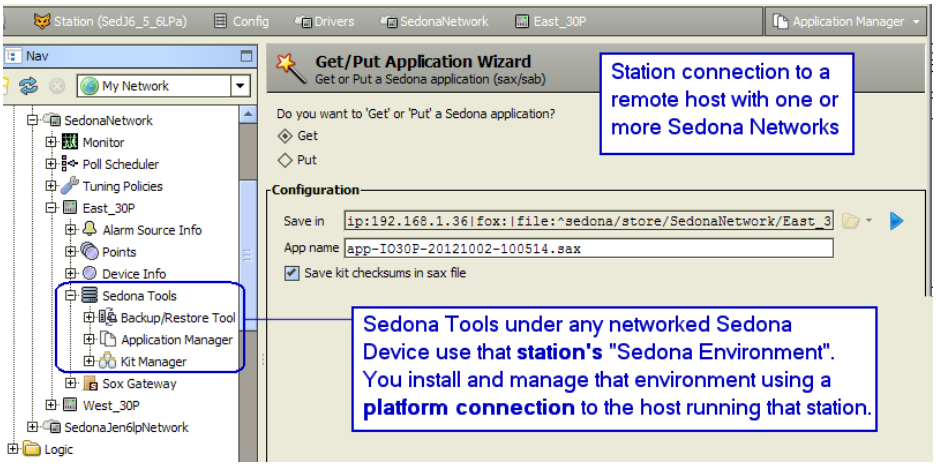
Note: *Procedures to get started with Sedona environment management are located in the "Sedona Framework Network Driver Installation" section, including subsections "JACE workflow" on page 1-2 and "Installing Sedona environment files in a JACE" on page 1-3.*

This section provides the rationale for remote Sedona environment management, along with reference details on the **`Sedona Environment Manager`** platform view, and includes the following subsections:

Sedona environment overview

Starting in AX-3.7 with Sedona Framework TXS-1.2, many Sedona TXS features can be "pushed down" from Workbench to the JACE controller level, allowing the JACE to be the central location for connecting to and provisioning Sedona devices. Included are Sedona device app saves (gets) and backups stored in the JACE *station's file space*, allowing a *station backup* or *save* to capture the *entire system*. Also, all necessary Sedona kits, manifests, and platform files can reside on the JACE, independent of Workbench.

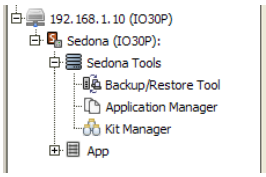
Figure B-1 *Sedona Device provisioning "through the JACE" example (Application Manager)*



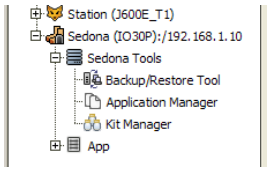
As shown in Figure B-1, each networked Sedona device has Sedona Tools that work in this manner. In addition, each Sedona device also has a "Sox Gateway" for access of the device's Sedona app.

Note: *This differs from Sedona device provisioning or app access when using a "direct Sox connection" from Workbench, or if using a "Tunneled Sox" connection through a JACE. Such connections are shown below.*

Direct Sox connection



Tunneled Sox connection

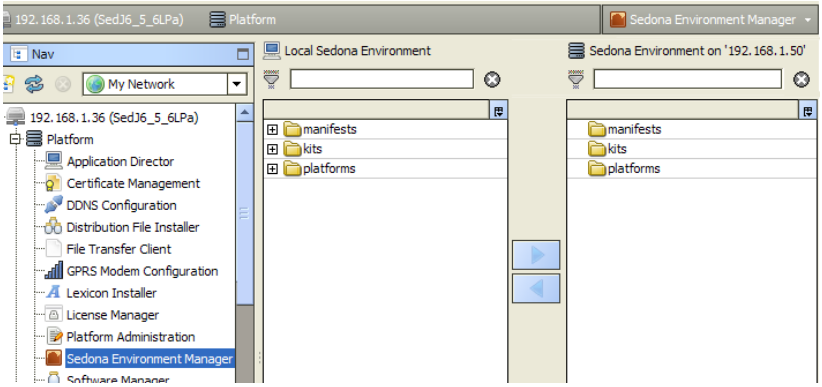


In either Sox session shown above, the Sedona Tools and App access use your local (Workbench) Sedona environment—that is, the Sedona kit manifests, kits, and platform database on your Workbench PC.

To support provisioning "through the JACE" (as in Figure B-1), the JACE requires "Sedona environment files" installed, using a `!sedona` folder with subfolders for Sedona kit manifests, kits, and platforms.

To install and manage this environment, you use a new *platform view* on the JACE: the **`Sedona Environment Manager`** (Figure B-2).

Figure B-2 *Sedona Environment Manager platform view for AX-3.7 JACE*



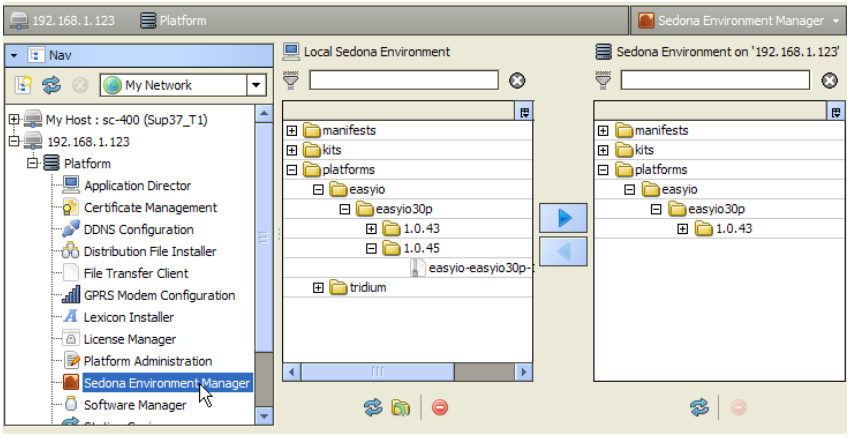
In this platform view, the *left-side* shows folders in your *local* (Workbench) Sedona environment. The equivalent Sedona environment folders on the *remote* (typically JACE) host are on the *right side*.

Sedona Environment Manager

The **Sedona Environment Manager** is a platform view for installing and managing the Sedona environment of any remote AX-3.7 or later host (typically JACE controller).

Note: *The* **Sedona Environment Manager** *platform view does not appear on a "localhost" platform connection, (e.g. with a Supervisor or Workbench engineering PC)—similar to other platform views like the Station Copier or File Transfer Client. In that case, you use Windows Explorer to manage your local Sedona environment, and/or Workbench's tool Sedona Installer (to import Sedona environment files). Also currently, the Sedona Environment Manager appears on a platform connection to any remote AX-3.7 or later host—including, for example, JACEs without any installed Sedona TXS modules (nsedona, sedonanet, etc.). In these cases you can safely ignore the Sedona Environment Manager.*

Figure B-3 *Sedona Environment Manager example in JACE platform connection*



Use this view to transfer (copy) Sedona manifest files, kit files, and platform database files between your Workbench PC's (local) Sedona environment and the Sedona environment of the remote host. Note that it works in *both directions*—for example, you can copy Sedona environment files *from* a connected JACE, adding them *to your local Workbench engineering PC or Supervisor*. This can be useful when different Workbench hosts are configuring various Sedona devices networked under the same JACE controller.

For a "quick-start" procedure for installing a Sedona environment on a JACE, see "Installing Sedona environment files in a JACE" on page 1-3. This rest of this section describes various aspects of this view, including some buttons and controls that are useful in the ongoing management of Sedona environments.

See the following for more details:

- "Loading Sedona Environment" on page B-3
- "View areas" on page B-4
- "Filter box" on page B-4
- "Mark missing files" on page B-4
- "Delete marked" on page B-5
- "Import Sedona files" on page B-5
- "Folder popup menu" on page B-5

Loading Sedona Environment

In a platform session, when you first access the **Sedona Environment Manager**, this tool "learns" both your local Sedona environment files and those in the remote host.

Figure B-4 *Loading Sedona Environment popup while the contents of both locations are learned*

A popup "Loading Sedona Environment" dialog (Figure B-4) appears during learning. Depending on the size of the two environments, loading can last only a couple of seconds, to several seconds. When loading is finished, you can navigate through both environments (sides) of the view to see what files are present.

At any time, click Workbench's **Refresh** button (on the toolbar) to reload the Sedona environments.

View areas

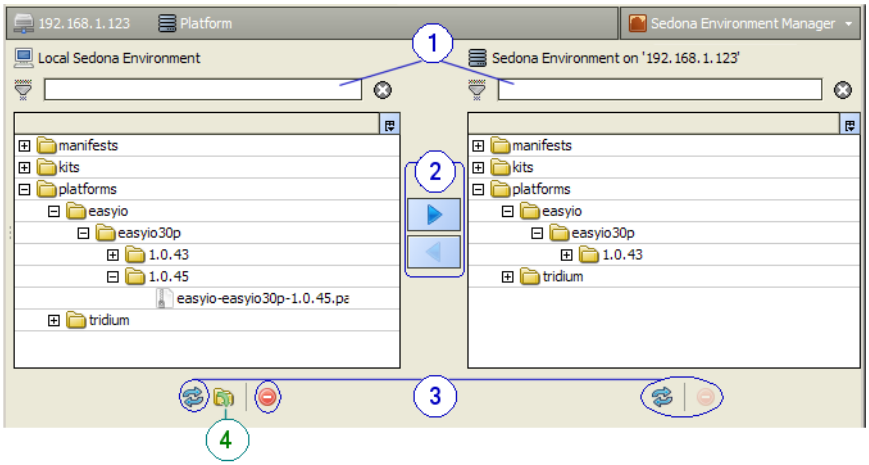
The *left* pane of the **Sedona Environment Manager** reflects your *local* (Workbench) Sedona environment—or more specifically, the contents of those three `!sedona` subfolders:

- `manifests`
- `kits`
- `platforms`

Note that other subfolders may exist in your Workbench Sedona environment—however, only the three subfolders above are reflected in this view.

The *right* pane of the **Sedona Environment Manager** reflects those Sedona environment folders on the *remote host* (typically JACE). However, if making a remote platform connection to a Supervisor, you can use this to manage its Sedona environment as well.

Figure B-5 *Sedona Environment Manager areas and common controls*



Both view panes offer identical controls and buttons, including:

1. A "filter box" at top (see "Filter box").
2. Directional "transfer" (arrow) buttons between the panes. One becomes active whenever you have one or more files marked. It shows the direction the file transfer will occur when you click it.
3. Bottom buttons are to globally **Mark missing files**, or to **Delete marked** items. See "Mark missing files" and "Delete marked" for more details.
4. Note only the left (local) side has a bottom **Import** button. See "Import Sedona files".

Additionally, you can right-click a folder in either pane for a popup shortcut menu. See "Folder popup menu" on page B-5.

Filter box

In either side, to filter (hide from display) any folder that does not contain a file named using a character string, click in the "filter box" and type at least one character. If needed, use the filter box in both Sedona environment sides.

Note: *This filter varies from some others in Workbench—it is* case-sensitive*, with "wildcards" assumed on each end.*

For example, if you are looking for a kit or manifest with a checksum of `50b48a1f` in its file name, type in `50b` or `48a1` or some other portion. When you expand the manifests or kits folders, only folders with a file name inclusive of that string are shown. Note this does *not* mark files—it only makes them visible.

Click the clear button at the end of a filter box to remove the filter (all folders display).

Mark missing files

In either side, to globally mark all Sedona environment files *missing on the other side*, click the button (Mark missing) at the bottom of the pane. If missing files are found, parent folders expand to show marked files, and the appropriate transfer (arrow) button is enabled.

As this works in both directions, it may be most useful to use it under the right-side (remote) environment, to update Niagara Workbench's Sedona environment with any files that only the JACE has. Otherwise, unless the JACE has enough available file space to receive the *entire* Workbench Sedona environment, you would likely want to unmark various folders before transferring files to the JACE.

Delete marked

In either side, if one or more files are marked, the button (Delete marked) at the bottom is enabled.

If you click it, a confirmation popup dialog appears.

Figure B-6 *Confirmation popup on delete marked*

Either click **Yes** to delete the marked file(s), or else click **No** to cancel the delete (files remain marked). After file(s) are deleted, the Sedona environment is reloaded.

Caution *Be careful with delete, particularly on your Local Sedona environment side! There is no undo.*

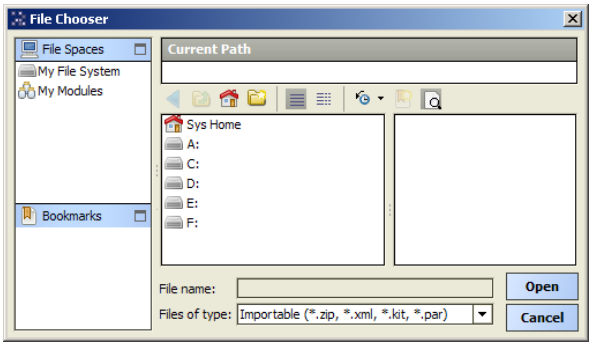
Import Sedona files

Under your local environment side (only), there is a button (Import) at the bottom of the pane. You can use this to import manifests, kits, or PAR files to the Sedona environment *of your Workbench PC* (only). You can also select a zip file, in which case it scans the contents of the zip file and imports any manifests, kits or PARs that it contains.

Note this same functionality exists in the **Sedona Installer** tool of Workbench. However, in some cases it may be convenient to run it from this view instead.

If you click this import button, a standard **File Chooser** popup dialog appears.

Figure B-7 *Standard file chooser dialog to import Sedona files*



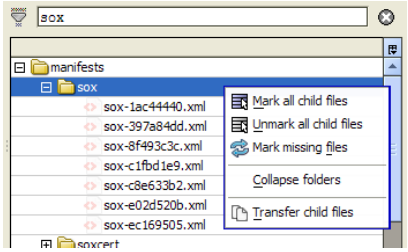
Navigate to your Workbench PC's location with the source file to select it, then click **Open** to import it.

Note: *After clicking* **Open** *to import, no other follow-up dialog appears, such as a confirmation, progress bar, or "finish" popup dialog (unlike when using the* **Sedona Installer***). Therefore, you may wish to look again at your local Sedona environment to make sure that new files were added.*

Folder popup menu

In either side, right-click any folder (at any hierarchical level) for a popup menu.

Figure B-8 *Popup menu for any folder (example)*



As shown in Figure B-8, menu choices are:.

- **Mark all child files** — Selects all files under this folder (and any of its subfolders).
- **Unmark all child files** — Deselects all files under this folder (and any of its subfolders).

- **`Mark missing files`** — Selects all files under this folder (and any of its subfolders) that are missing *on the other side's* Sedona environment.
- **`Collapse folders`** — Collapses this folder (marked files remain marked).
- **`Transfer child files`** — Transfers *all* files under this folder (and any of its subfolders).

Note: *On either side, there is also a "global"* **`Mark missing files`** *button at the bottom of the pane. It is equivalent to choosing the right-click "Mark missing files" option for each of the top-level folders.*

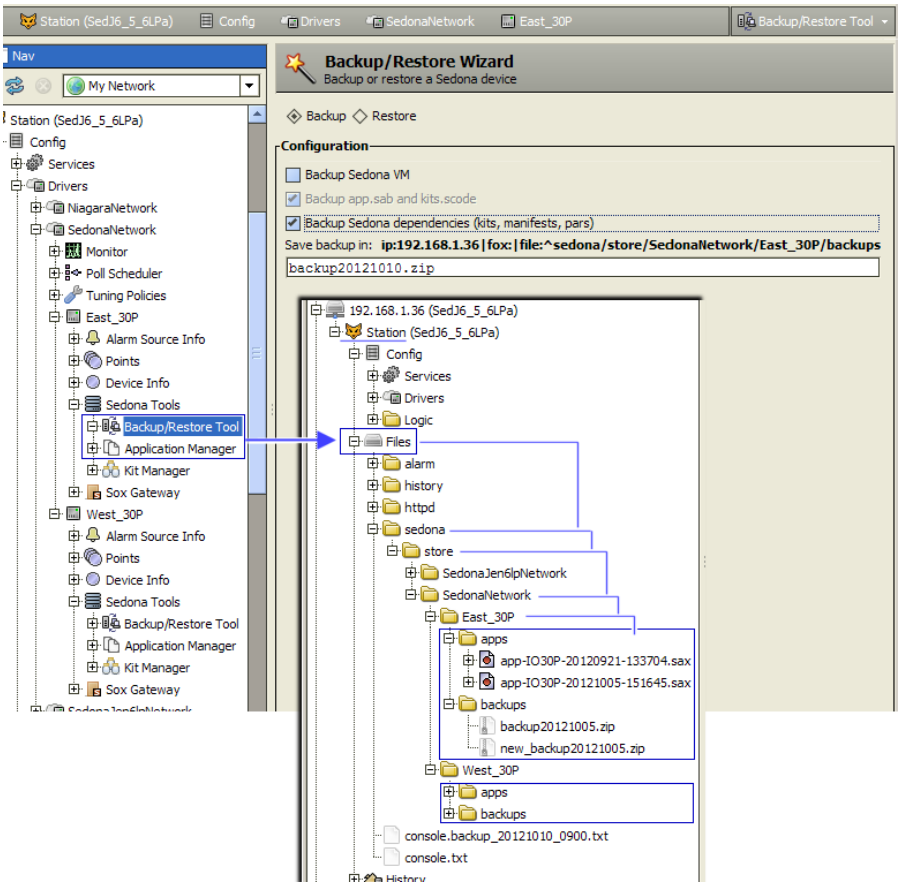
Station backup notes

A Niagara station backup of a JACE with an installed Sedona environment includes all its environment files, plus all saved Sedona apps and Sedona device backups. However, to ensure device backups are included you must edit its station's BackupService from defaults.

About Sedona device backups and app gets

Using the Sedona Tools under each networked Sedona device in a JACE station (or Supervisor station), you can backup each Sedona device, as well as "get" the app running in the device. For related details, see the *Sedona Framework TXS Sedona Tools Guide*.

Figure B-9 *Sedona Tools under a networked Sedona device save backup and app files to station's file space*



As shown in Figure B-9 above, when you perform such a Sedona device backup (using the Backup/Restore Tool) or "get" a device's app (using the Application Manager), the resulting backup.zip file or app.sax file is saved in that running *station's file space*, in the following folders:

- backup
  `^sedona/store/`*`SedonaNetwork`*`/`*`SedonaDevice`*`/backups/backup`*`YYYYMMDD`*`.zip`
- app
  `^sedona/store/`*`SedonaNetwork`*`/`*`SedonaDevice`*`/apps/app-`*`YYYYMMDD-hhmmss`*`.sax`

Similarly, you can use the Sedona Tools under a networked device to *restore* such a saved backup, or "put" such a saved app back on a device.

Therefore, a JACE *station backup* can capture the *complete configuration* of that JACE controller, *including all Sedona devices* networked under it (as well as that JACE's Sedona environment files).

Note: *To ensure that Sedona device backups are included, you must edit* **some BackupService** *properties in the host's station from default values. See "Editing the station's BackupService".*

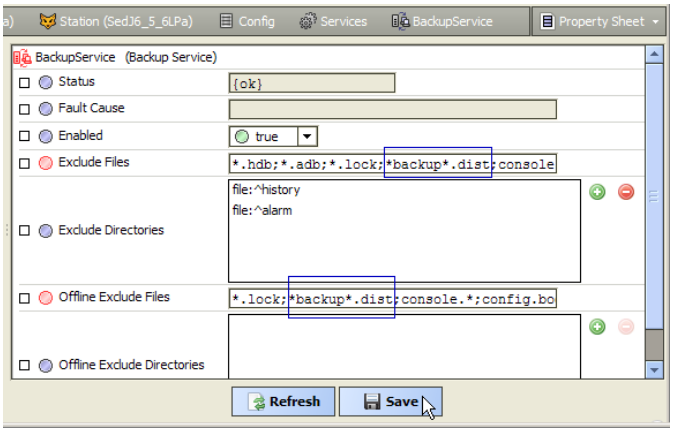
Editing the station's BackupService

Perform the following on the BackupService of any AX-3.7 or later station running any type of Sedona Network to ensure Sedona device backups are collected.

To edit the station's BackupService to collect Sedona device backups

Step 1 Using Workbench, open the station and expand the **Config** > **Services** node.

Step 2 Open the BackupService property sheet.
(right-click **BackupService**, select **Views > Property Sheet**).



Step 3 For both properties Exclude Files and Offline Exclude Files, *change* the portion of string value from: `*backup*` to be instead: `*backup*.dist` (as shown above), and click **Save**.

This will allow default Sedona device `backup`*`YYYYMMDD`*`.zip` files to be backed up. In the case of a Supervisor station, this also prevents Niagara provisioning backups from being (redundantly) backed up.

Step 4 Save the station.
(right-click the station in the Nav tree and select **Save Station**).

For related details on a station's BackupService, refer to *NiagaraAX User Guide* sections "backup-BackupService" and "backup-Backup Manager".

Appendix C

Sedona users and roles management

Starting in Sedona TXS-1.2, station components and views were added that provide a centralized point of management for Sedona users in a station's networked Sedona Framework devices.

Note that a Sedona Framework device app has always provided a "users" UserService component (from `sys` kit) with a default **`Sedona User Manager`** view to manage users in the app. Properties of each user determine their Sox login credentials to the device, Sedona component permissions in the app (across four groups), and Sox provisioning permissions (across three different areas).

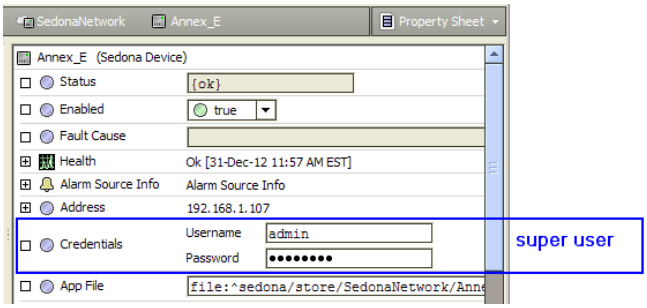
However, management of users across many devices required one-by-one Sedona app access in each device. This could be a time-consuming and potentially error prone task.

The following main sections provide more details:

About the device super user

Niagara integration of a Sedona Framework device requires it to have a defined "super user" in its app. This user has all possible permissions and provisioning capabilities, and is sometimes referred to as the "admin user". This is the user you reference when you add a Sedona device component to the station's Sedona network, entering the Username and Password in the "Credentials" property fields (Figure C-1).

Figure C-1 *Sedona device's credentials must reference Sedona super user*



Note that when any Sedona device is networked under the station, that Niagara ultimately *controls* this super user account—using this account for everything from managing points, doing provisioning, and providing access to the device's app via the Sox Gateway. This also includes user synchronization from the station's Sedona user management views. Therefore it is important that this user is strictly controlled.

A Sedona network's **`Sedona Device User Manager`** view provides a special function for globally updating this super user in all devices in the network. Usage does *not* require any Sedona users or user roles to be defined under the station's Sedona User Management Service ("top tier" architecture).

In fact, in a typical Niagara integration of Sedona Framework devices, where access to Sedona device apps and/or Sedona device provisioning is performed "through the station", the *only* user of any consequence is each device's referenced super user. Other Sedona users, including those created with users and roles in the station's service and then mapped to devices in the network view, apply only when accessing devices using direct (or "tunneled") Sox connections.

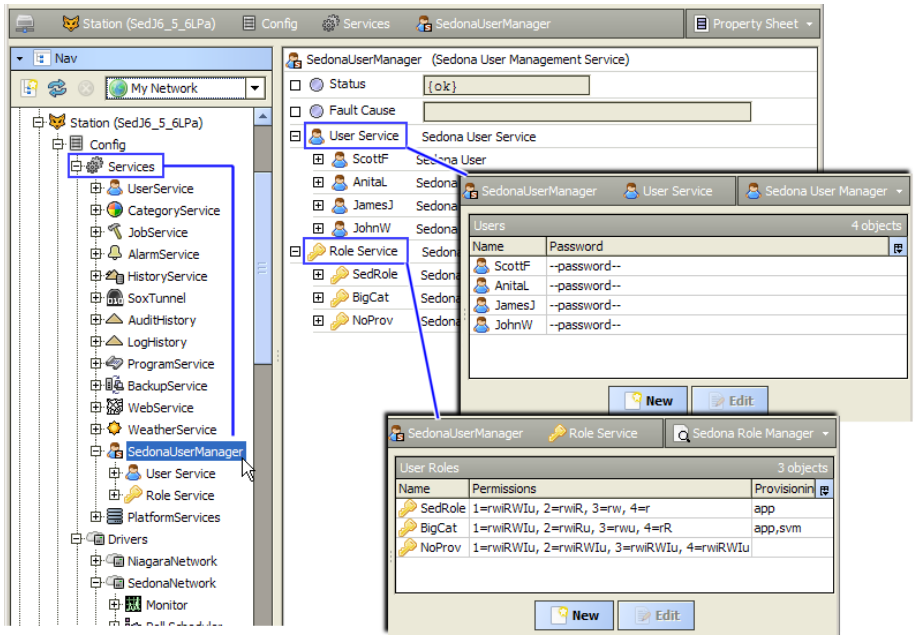
Sedona user management overview

Starting in Sedona TXS-1.2, you now have a "centralized" method to add, delete, and modify Sedona users and their roles in Sedona Framework devices networked under a station. Additionally (and typically most useful) is the ability to *globally change* the user name and password of the single "super user" of each networked Sedona device.

Station architecture for Sedona user management

Station architecture for Sedona user management uses a 'tiered" approach, where:

- The "top tier" is in the station's Services container, where a SedonaUserManagementService (**`Sedona User Manager`**) provides child "UserService" and "RoleService" components.

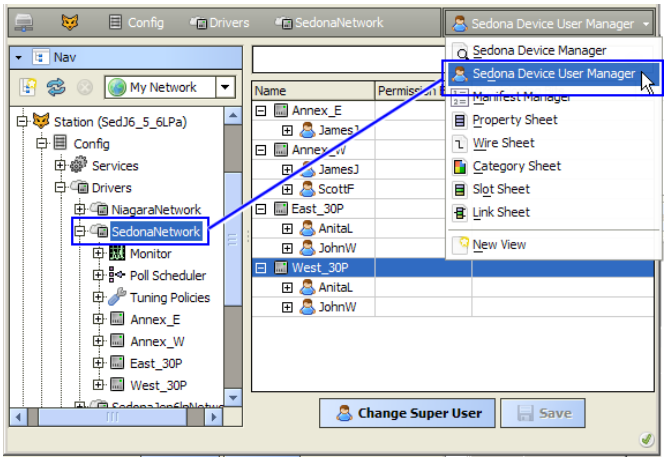


As shown above, each of these two child services has a default manager view.

- In the User Service's manager view, you create/manage users (name and password).
- The Role Service's manager view lets you create/manage roles (permissions and provisioning rights).

Users and roles can apply to all Sedona networks in the station. By themselves, components in this tier do not do much, at least until they are mapped to networked devices in the next tier.

- Any Sedona network in the station (SedonaNetwork, SedonaJen6lpNetwork) now has a **`Sedona Device User Manager`** view.



In this view, all networked child Sedona devices appear. For any device, you can map (add or delete) users and roles defined in the station's service, where each user is assigned to a role. This allows the same user to have different roles (and therefore permissions) in different devices in the network.

This network view also provides a "Change Super User" function, to globally modify the super user account used by Niagara to communicate to *all* Sedona devices in the network (the account referenced in the "Credentials" property of each SedonaDevice or SedonaJen6lpDevice). This user account is not seen in the station in any Niagara manager view, but is "owned" by Niagara and used for almost all communications in a Niagara integration. This function can be handy if many devices were installed with factory-default super user credentials, preventing against unauthorized access.

Changing the Sedona super user in all devices

This is an online function only. Note the following prerequisites:

- All devices in the Sedona network must have an "ok" status (cannot be down, disabled, or in fault).
- No Sedona user with the entered Username already exists in a device's app.
  The only exception: if the entered Username matches the name of the current super user. In that case, only the password for that user is updated.

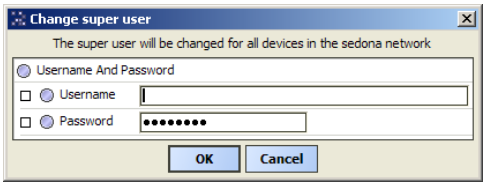
To change the Sedona device super user for all networked Sedona devices

Step 1 Ensure the station has a **SedonaUserManager** (SedonaUserManagementService) in its **Services** container.

- If present, go to Step 2.
- If not present, open the sedonanet palette and drag the **SedonaUserManager** into the station's **Services** container (no further configuration is necessary).

Step 2 Right-click the Sedona network and select **Views > Sedona Device User Manager**.

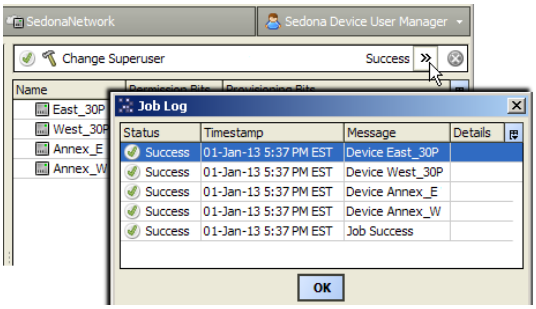
Step 3 In the **Sedona Device User Manager** view, click the **Change Super User** button.



Step 4 In the **Change super user** popup dialog (above), enter the name and password for the super user.

Note: *User name must be from 1 to 7 alphanumeric characters, is case-sensitive, and must begin with a letter and have no spaces or punctuation, apart from underscore (_).*

Step 5 Click **OK**. A "Change Superuser" job is initiated, with a progress bar at the top of the view. The job completes showing either "Success" or "Failed".



Click the » details button for a popup **Job Log** dialog, as shown above.

How it works: After the new super user is added to a device, the old super user is deleted from the device and the device's app is saved. The "Credentials" properties for each Sedona device component in the network is automatically updated with the new super user name and password.

Note: *If this function fails for a device for some reason, it is logged in the job log, and the next device is tried. Therefore, the super user for the maximum number of devices is updated.*

Centrally managing Sedona users

Centrally managing Sedona users in networked Sedona devices is a three-part process:

1. Create or modify Sedona *users* in the station's **SedonaUserManager** > **User Service**.
2. Create or modify Sedona user *roles* in the station's **SedonaUserManager** > **Role Service**.
3. Map users and roles to *devices* in a Sedona network's **Sedona Device User Manager** view.

For an overview of how this looks in the station, see "Station architecture for Sedona user management" on page C-2.

After users and roles are mapped to devices, any changes (modifications, deletions) to users or roles result in automatic *synchronization* of those items to mapped devices. For example, if you change the password for a Sedona user in the **Sedona User Manager** view, and/or change the permissions in the **Sedona Role Manager** view for a role, those changes are automatically synchronized in the Sedona app of mapped devices.

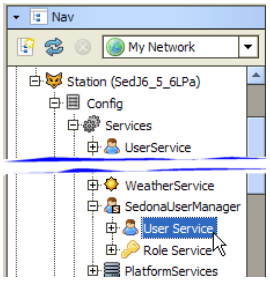
Step-by-step procedures and associated information are in these sections:

Creating and modifying Sedona users and roles

A prerequisite is the station has a **SedonaUserManager** (SedonaUserManagementService) in its **Services** container. If not present, open the sedonanet palette and drag the **SedonaUserManager** into the station's **Services** container.

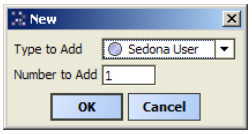
Creating new Sedona users

Step 1 Expand the station's **Services** container to reveal the **SedonaUserManager** > **User Service**.

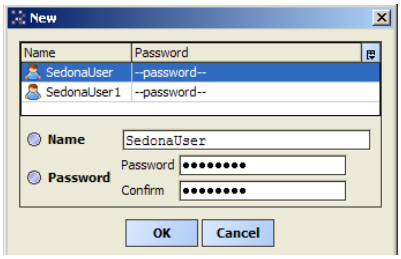


Step 2 Double-click the **User Service** for the **Sedona User Manager** view.

Step 3 In the Sedona User Manager, click **New**. A popup **New** dialog appears.



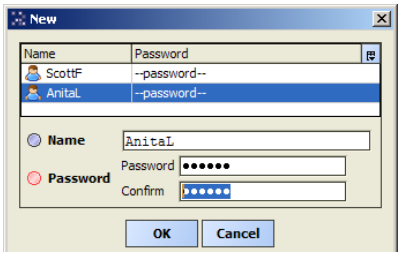
Step 4 In the **New** dialog, either accept 1 as the number to add, or type in another number to add multiple users. Click **OK**. A second **New** dialog appears.



Step 5 For each user (two shown above), change the default (and invalid) "SedonaUser" to a valid name for the resulting User component(s) to be created later in Sedona apps, and also enter a password.

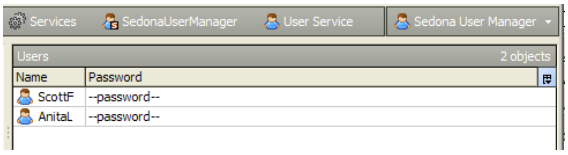
Note: *Name must be from 1 to 7 alphanumeric characters, is case-sensitive, and must begin with a letter and have no spaces or punctuation, apart from underscore (_). These naming restrictions reflect Sedona component naming guidelines. If you try entering names that fall outside these guidelines, an error results, and the users are not added.*

If creating multiple users in this dialog, click to select (highlight) each user to edit Name and Password.



Step 6 After naming the user(s) and assigning passwords, click **OK** in the **New** dialog.

The user or users are added, as shown below.



Step 7 If needed, add additional new users by clicking **New** in the **Sedona User Manager** (Step 3).

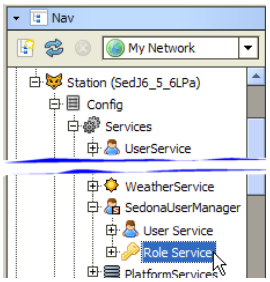
You can also edit (modify) Sedona users by double-clicking them in the **Sedona User Manager**, which produces an **Edit** popup dialog, identical to the second **New** popup dialog. In the **Edit** popup dialog, you can modify both name and password for any user.

You can also delete Sedona users, via the right-click shortcut menu on any selected user.

Note: *If modifying or deleting a user that has been mapped to one or more networked Sedona devices, this results in a synchronization task to make corresponding changes in those device's Sedona app. For related details, see "User Synchronization" on page C-13.*

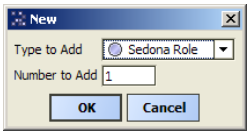
Creating new Sedona roles

Step 1 Expand the station's **Services** container to reveal the **SedonaUserManager** > **Role Service**.

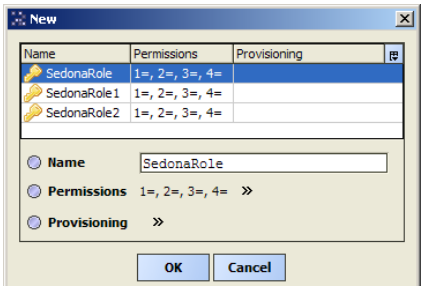


Step 2 Double-click the **Role Service** for the **Sedona Role Manager** view.

Step 3 In the Sedona Role Manager, click **New**. A popup **New** dialog appears.



Step 4 In the **New** dialog, either accept 1 as the number to add, or type in another number to add multiple roles. Click **OK**. A second **New** dialog appears.

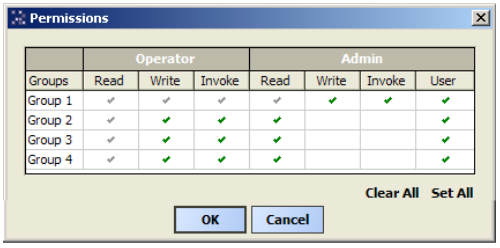


Step 5 For each role (three shown above), optionally change the default name "SedonaRolen" to a another more descriptive and unique name.

Note: *Sedona name restrictions do not apply to user roles, as roles do not have direct component representation in a Sedona app. Therefore, Niagara component names over 7 characters are allowed.*

Wait until *after the next two steps* before clicking OK.

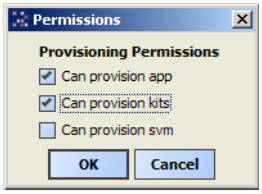
Step 6 For each role, click the **Permissions** » control for a separate **Permissions** popup dialog, and click in cells to toggle (set or clear) permissions in the four Sedona component groups.



Initially (if a new role), all component group permissions are cleared. As needed, click on the "**Clear All**" or "**Set All**" text to clear or set *all* permissions. Then continue to click in individual cells to toggle.

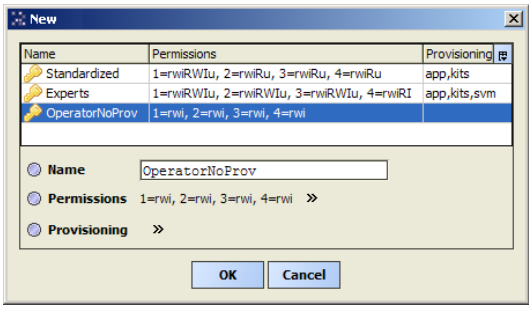
Click **OK** to set the role's component permissions and return to the previous **New** dialog (and to the next step, to set provisioning permissions).

Step 7 For each role, click the **Provisioning** » control for a separate **Provisioning** popup dialog, and click in check boxes to toggle (set or clear) permissions for the three provisioning types.



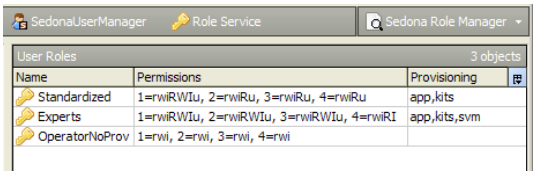
Initially (for any new role), all provisioning permissions are cleared. The example role above is being configured to all app provisioning (Get and Put via the Application) and kits provisioning (via **Kits Manager**), but not provisioning changes to the Sedona virtual machine (SVM).

Click **OK** to set the role's provisioning permissions and return to the previous **New** dialog.



Step 8 After reviewing names for the role(s) and assigning each role component **Permissions** and **Provisioning** permissions (see example shown above), click **OK** in the **New** dialog.

The role or roles are added, as shown below.



Step 9 If needed, add additional new roles by clicking **New** in the **Sedona Role Manager** (Step 3).

You can also edit (modify) Sedona roles by double-clicking them in the **Sedona Role Manager**, which produces an **Edit** popup dialog, identical to the second **New** popup dialog. In the **Edit** popup dialog, you can set or clear permissions on any of the three provisioning types.

You can also delete Sedona roles, via the right-click shortcut menu on any selected role.

Note: *If modifying or deleting a role that has been mapped to one or more networked Sedona devices, this results in a synchronization task to make corresponding changes in those device's Sedona app. For more details, see "User Synchronization" on page C-13.*

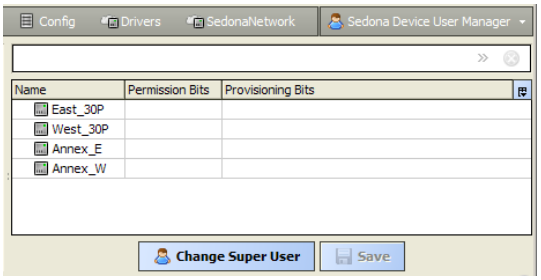
Mapping users and roles to networked Sedona devices

A prerequisite is that at least one Sedona user and Sedona user role has been created under the station's **SedonaUserManager** (SedonaUserManagementService). See "Creating and modifying Sedona users and roles" on page C-4.

Once Sedona users and roles exist in the station, you can map them in combinations to Sedona devices networked under any Sedona network in the station (SedonaNetwork, SedonaJen6lpNetwork). You do this using the available **Sedona Device User Manager** view on each Sedona network, with operations described as follows.

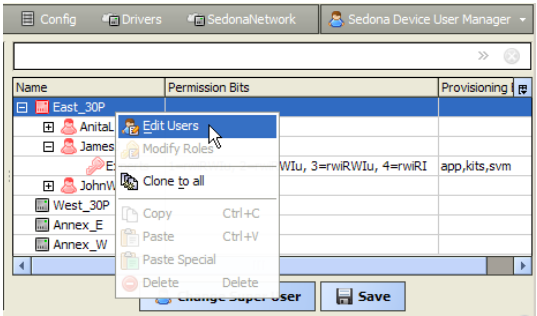
Accessing and using the network's Sedona Device User Manager

Step 1 Right-click the Sedona network (in the Nav tree or in its property sheet) and select **Views > Sedona Device User Manager**.



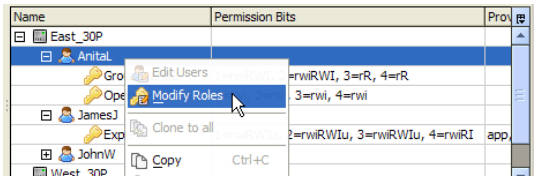
All networked Sedona devices are listed as root elements in a "tree table", where initially no device has any users (with user roles) assigned to it, similar as to shown above.

Step 2 Map users to devices by *right-clicking* on any selected node in the tree table for various menu options.



Unsaved changes appear as *red nodes* in the tree table—note the **Save** button becomes available.

Right-click menus vary by selected tree node. As shown above, right-clicking a device shows a menu with **Edit Users**, but if right-clicking a saved user, a **Modify Roles** option appears (see below).



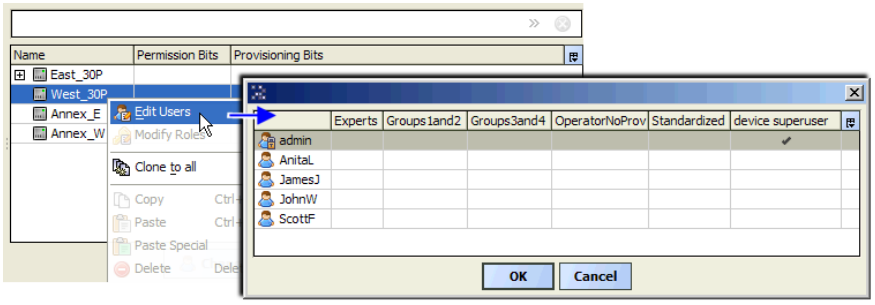
If right-clicking a role, only a **Delete** menu option appears (not shown here).

Various other right-click menu options can be used when mapping users and user roles to devices in this view, which operates *differently* than most Workbench manager views.

Remember to **Save** when done making changes in this view. Back to "Mapping users and roles to networked Sedona devices".

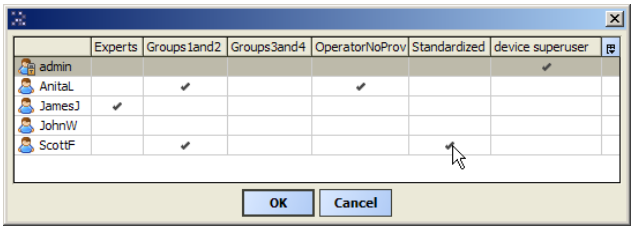
Adding users and roles to a Sedona device

Step 1 In the network's **Sedona Device User Manager**, right-click a device and select **Edit Users** (or with a device selected, click the edit user tool on the toolbar).



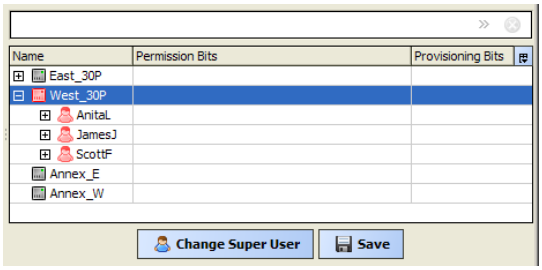
As shown above, a popup dialog lists all available user in rows, and all available roles as columns. Initially, no users have roles assigned, except for the current "device superuser" (fixed).

Step 2 Click in cells to select (or toggle/deselect) roles for users.

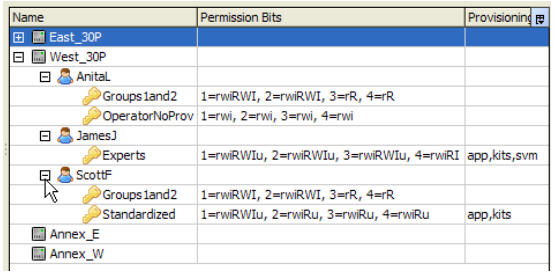


Note that role permissions are "OR'ed" together, that is they accumulate. As shown above, the two users AnitaL and ScottF are being given permissions for two roles.

Step 3 Click **OK** to close the dialog and accept the changes.



Note changes are still *uncommitted*—click the **Save** button in the view to commit.

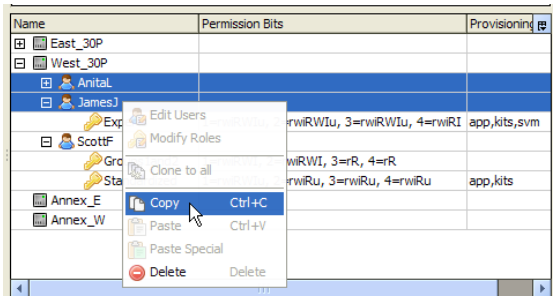


As shown above, when you expand users under the device you see their assigned roles, including the corresponding permission bits for the component groups 1-4 and provisioning permissions.

Back to "Mapping users and roles to networked Sedona devices" on page C-7.

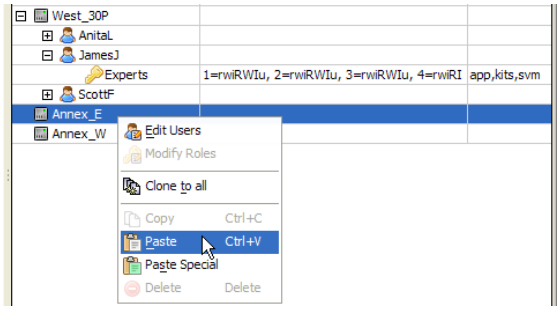
Copying Sedona users from device to device

Step 1 In the network's **Sedona Device User Manager**, right-click a user and select **Copy** (or Ctrl-click to select multiple users, then right-click to select **Copy**, or select the tool on the toolbar).

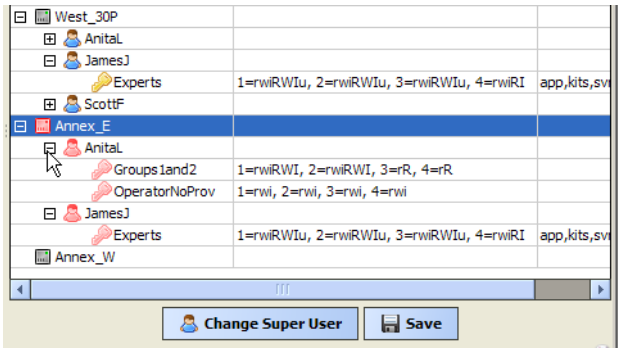


As shown above, two users have been selected and copied.

Step 2 Right-click the target device, and select **Paste**.



Now the user(s) copied are under the target device (in this case, device Annex_E).



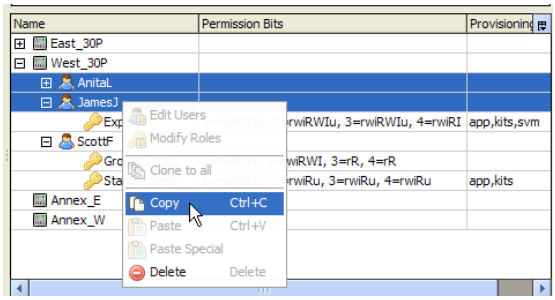
Step 3 Note changes are still *uncommitted*—click the **Save** button in the view to commit.

Back to "Mapping users and roles to networked Sedona devices" on page C-7.

Copying Sedona users as an overwrite (Paste Special)

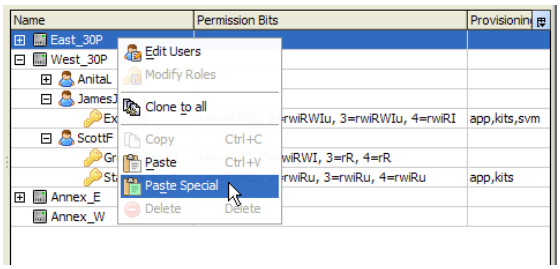
Sometimes you may want to *replace* the current Sedona user configuration in selected device(s) with the user(s) currently copied. You can do this using the **Paste Special** menu option on the target device(s). It is equivalent to *deleting* all current users in the device(s), then adding the copied users.

Step 1 In the network's **Sedona Device User Manager**, right-click a user and select **Copy** (or Ctrl-click to select multiple users, then right-click to select **Copy**, or select the tool on the toolbar).

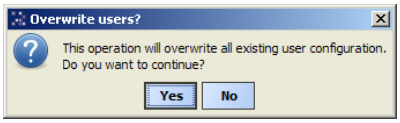


As shown above, two users have been selected and copied.

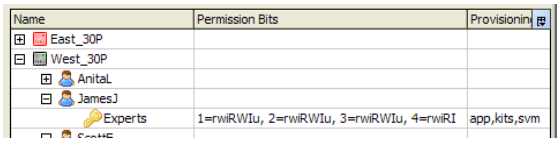
Step 2 Right-click the target device or devices, and select **Paste Special**.



A popup confirmation dialog appears, warning that all current user configuration will be overwritten.



Click **Yes** to accept (or else **No** to cancel).



Step 3 Note changes are still *uncommitted*—click the **Save** button in the view to commit.

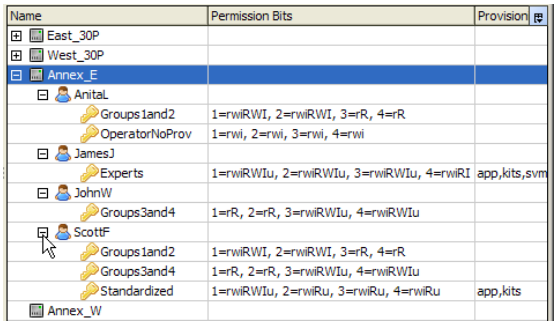
Back to "Mapping users and roles to networked Sedona devices" on page C-7.

Cloning users and roles to all devices

Sometimes you may want the *identical* user and role configuration in *all* networked devices. You can do this by configuring one device, then selecting the **Clone to all** menu option. It is equivalent to *deleting* all users in *all other* devices, then adding the users and roles in the selected device.

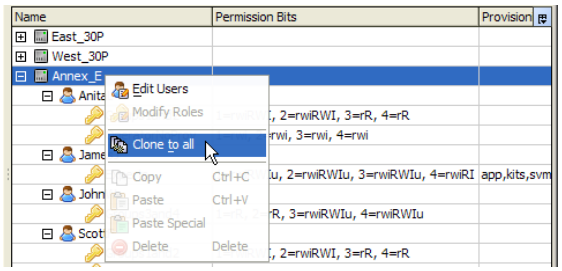
Even if not the final user configuration, it may be a starting point to further tweak users and roles in other devices as needed (for example to selectively delete users or roles, or to specify additional users and roles).

Step 1 In the network's **Sedona Device User Manager**, expand a device and all its users to see all roles.

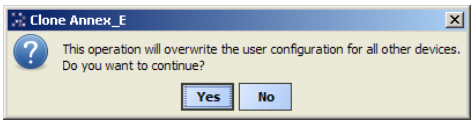


Verify this is the user configuration you want in all networked devices. If needed, make any changes.

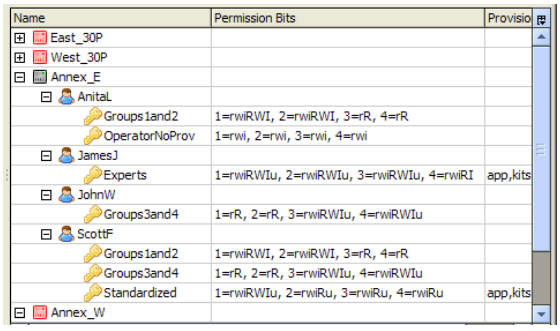
Step 2 Right-click the device and select **Clone to all** (or with the device selected, click the clone to all tool on the toolbar).



A popup confirmation "**Clone *DeviceName***" dialog appears, warning that the current user configuration in all other devices will be overwritten.



Click **Yes** to accept (or else **No** to cancel).



The figure above shows changes now marked in the other networked devices.

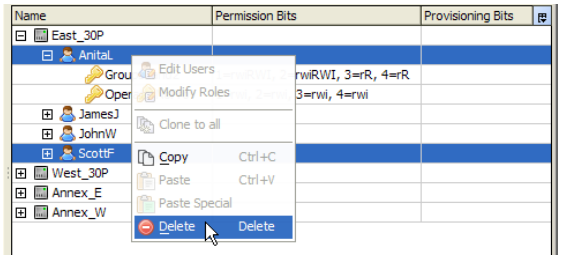
Step 3 Note changes are still *uncommitted*—click the **Save** button in the view to commit.

Back to "Mapping users and roles to networked Sedona devices" on page C-7.

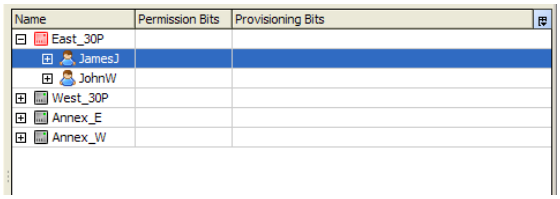
Deleting users and roles

Working in the tree table of the **Sedona Device User Manager** view, you can selectively delete users and user roles from devices.

Step 1 To delete a user, right-click and select **Delete** from the menu, or Ctrl-click to select *multiple* users and then right-click and select **Delete**.

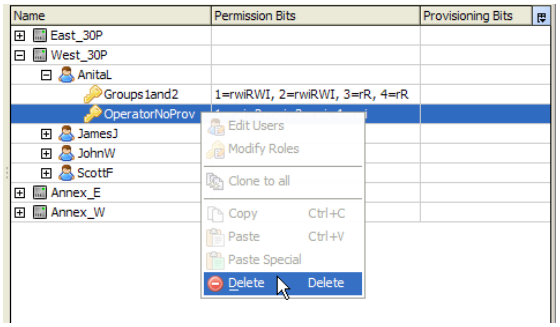


The selected user(s) are removed from the tree and the device appears red (to indicate change).

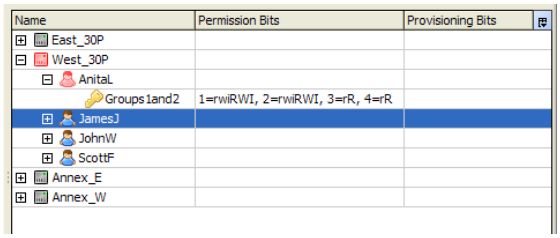


Step 2 Note changes are still *uncommitted*—click the **Save** button in the view to commit.

Step 3 To delete a role, expand a user under a device and right-click a role and select **Delete** from the menu, or Ctrl-click to select *multiple* roles and then right-click and select **Delete**.



The selected role(s) are removed from the tree and the user appears red (to indicate change).



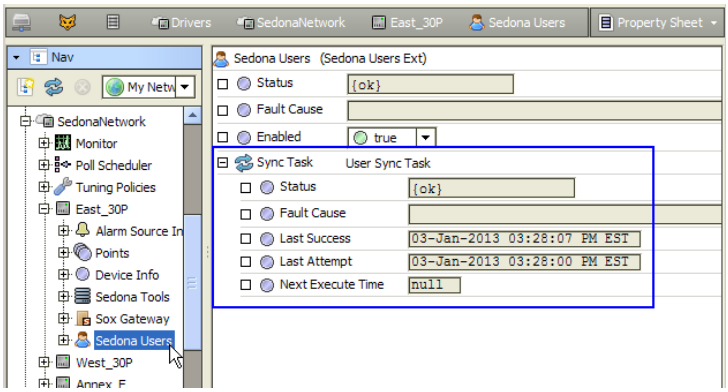
Step 4 Again, note changes are still *uncommitted*—click the **Save** button in the view to commit.

Back to "Mapping users and roles to networked Sedona devices" on page C-7.

User Synchronization

When the **SedonaUserManager** service (SedonaUserManagementService) ia added to the station, it creates and registers a **Sedona Users** device extension (SedonaUsersExt) under each Sedona device in all Sedona networks.

Figure C-2 *Sedona Users device extension handles user synchronization from station to device's app*



Internally, the extension stores the current mapping of users and roles for the device, and it also manages synchronizing those mappings out to the Sedona app in the device.

The extension's **Sync Task** (UserSyncTask) component performs the synchronization work. It keeps track of when synchronization is necessary, performs the work to synchronize with the Sedona device.

Typically you do not need to access this component; however it may become useful for debugging purposes. If necessary, you can force a device to synchronize its users via a right-click **Execute** action, which results in an immediate synchronization attempt.

User synchronization details

There are three conditions that cause user synchronization for a device to be executed. When any of the following conditions are met, it causes the SedonaUsersExt to *schedule* a sync task with its UserSyncTask component.

1. When you save user or role mappings in the **Sedona Device User Manager**, this saves those mappings to the SedonaUsersExt of each affected device. A sync task is scheduled for those devices.
2. When you delete or modify a user in the station's Sedona User Service. All devices that have that user mapped are scheduled for user synchronization.
3. When you delete or modify a role in the station's Sedona Role Service. All devices that have a user with that role mapped are scheduled for user synchronization.

Note by *schedule*, this means the sync does not happen immediately. Instead, there is a brief delay in case more changes are about to be made. The "Next Execute Time" property of the UserSyncTask (if non-null), indicates a sync is pending, showing when the sync will occur.

In case a sync task fails, it automatically reschedules itself to retry at a later time. This state is maintained, even across station restarts. The sync task will not stop trying until it either succeeds, or else you manually invoke the **Cancel** action on the UserSyncTask component.

Additional notes on user synchronization Additional facts on user synchronization include:

- If the parent Sedona device is down, disabled, or in fault, the UserSyncTask immediately fails and reschedules itself for a later execution.
- The UserSyncTask *never* synchronizes the device's super user. Use the "Change Super User" function in the Sedona Device User Manager for this. See "About the device super user" on page C-1 and "Changing the Sedona super user in all devices" on page C-3 for related details.
- Synchronization always makes the app in the remote Sedona device match the user/role mappings stored in the Niagara station. This means any existing Sedona users that are not in the networked device's user/role mapping will be *removed* (except for the super user). There is no way to sync *from* a Sedona device up to the device's SedonaUsersExt in the station.