Technical Document

Niagara^{AX} Sedona Framework
Chopan Usage

November 9, 2011

Sedona Framework Chopan Usage

Confidentiality Notice

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

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

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

Trademark Notice

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

Copyright and Patent Notice

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

CONTENTS

PREFACE

Preface

This document explains the usage of CHoPAN (Compressed HTTP over Personal Area Network), an available protocol in the Sedona Framework TXS (Sedona Framework 1.1). For ease of reading, and also to match various component and view names in the NiagaraAX and Sedona Frameworks, "Chopan" replaces CHoPAN in many areas of this document.

Currently, Chopan applies only to wireless (Jennic-based) Sedona Framework devices networked using a QNX-based JACE-2, -6, -7 or JACE-x02 XPR controller with an installed "Sedona Jennic" option card.

This preface has the following sections:

- About this document
- Chopan FAQs
- Chopan usage in Sedona Framework terms
- Document change log

About this document

As noted earlier in this Preface, this document applies to the usage of Chopan in the Sedona Framework TXS 1.1 and later, and has the following main sections:

- Chopan software and hardware requirements
  Explains the NiagaraAX and Sedona Framework platform, software, and licensing requirements.
- Sedona Framework Chopan Quick Start
  Provides several step-by-step procedures for getting started with using Chopan.
- About Chopan in the Sedona Framework
  Provides background on why CHoPAN is used, with additional Chopan-related topics that apply to Jennic-based devices, including the engineering of them using NiagaraAX Workbench.

Chopan FAQs

Below are some frequently asked questions (FAQs) about Chopan usage with Sedona Framework devices.

Q: **What is Chopan?**

A: Chopan is a Tridium-developed protocol running over UDP/IP that is specifically designed to allow peer-to-peer data sharing between 802.15.4 wireless devices and/or a JACE controller.

Q: **Why use Chopan?**

A: For point value updates, Chopan provides bandwidth efficiencies over Sox in wireless (Jennic-based) networks of devices, which have notably less bandwidth than networks of Ethernet- and/or WiFi-equipped Sedona Framework devices. Chopan is also the only way that Jennic-based devices can directly share data, peer-to-peer. Further, if a Jennic-based device is configured to "hibernate" (noting Sedona Framework support for this is not widely available now), client "Chopan point" usage is *required* for point value updates in the NiagaraAX station—*instead* of Sedona proxy points. Such devices also use Chopan for other routines, such as "Maintenance Mode".

Q: **Why not use Chopan?**

A: While using a Chopan tuning policy for Sedona Network proxy points is similar to using Sox, using the client interface for Sedona devices involves configuration interfaces that may be abstract and unfamiliar to those familiar with NiagaraAX drivers. Chopan does not use authenticated sessions and is less secure than the Sox protocol.

***Q:* Is there an “easy way” to benefit from Chopan if I *don’t* have hibernating devices or *don’t* need to have devices share data directly, peer-to-peer?**

A: Yes: simply configure Jennic-based devices as Chopan *servers*. Then in the JACE station, for Sedona proxy points for those devices, specify tuning policies that use the “Comm Type” of “Chopan”. No Chopan client configuration of devices is required.

***Q:* How does Chopan compare to Sedona Framework 1.0?**

A: Sedona Framework 1.0 uses Sox protocol for all communications. The JACE can poll or subscribe to data within a Sedona Framework device, but all data sharing must go through the JACE.

The Sox protocol is designed around an authenticated session between a client and server. Each connection requires multiple messages to set up and tear down and dedicated resources to maintain. This makes it unsuitable for general data sharing between devices.

***Q:* Does peer-to peer mean I don’t need a JACE?**

A: Every network of Jennic-based devices still requires a coordinator. This is an 802.15.4 networking requirement. At this point, the only coordinators offered for deployment are the “Sedona Jennic” option card running in a JACE, or the USB Jennic adapter (coordinator) for the Sedona Framework.

Note that the USB coordinator is intended for use in a developer scenario only, from Workbench to a *single* Sedona Framework device. There are no plans to support it as the coordinator to a “network of devices”—for example using it with a AX SoftJACE or Windows-based JACE environment.

Chopan usage in Sedona Framework terms

The following is a list of terms and abbreviations used in this document when describing Chopan usage in the Sedona Framework. For other Sedona Framework terms, see “Sedona Terms” in the *NiagaraAX Sedona Networks Guide*. For general NiagaraAX terms, see the Glossary in the *User Guide*. Note that this glossary may grow over time, or may 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. The Sedona Framework 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 Framework components and services, including links between them, and all configuration properties. Components and services are selected from kits installed in the device.

Chopan Or CHoPAN, for Compressed HTTP over Personal Area Networks. It is a Tridium proprietary protocol used in Sedona TXS to support Jennic-based devices, particularly any hibernating device. Chopan runs over UDP/IP and is a “session-less” protocol (unlike Sox), offering a number of advantages over Sox in certain applications. Chopan usage involves different components and views in the NiagaraAX and Sedona Frameworks, and is the subject of this document.

ChopanService The component in the Jennic-based device’s app that provides communication capability over CHoPAN. By convention, it is located in the app’s “service” folder.

ChopanServlet The component in the Jennic-based device’s app that handles incoming CHoPAN point reads and writes (server-side functionality). By convention, it is a child of the ChopanService component.

Note that equivalent functionality in the JACE station is provided by the ChopanServer child of the station’s SedonaJen6lpNetwork component.

ChopanNetwork The top level component in the Jennic-based device’s app that can initiate reads and writes (client-side functionary). By convention, it is located in the app’s “service” folder. It can contain child ChopanDevice components.

ChopanDevice A component in the Jennic-based device’s app that is a proxy representation of a remote CHoPAN *server* device, providing addressing information. It can contain child Chopan point components. The device may represent another Jennic-based device or the JACE controller (coordinator).

Chopan point A component in the Jennic-based device’s app acting as Chopan proxy point, either of type **`ChoBoolPt`**, **`ChoBoolWr`**, **`ChoFltPt`**, **`ChoFltWr`**. You link these points to other components in the device’s app to share data.

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.

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 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 "hibernates" (sleeps) the majority of time, periodically "waking up" for short periods to execute routines and exchange data with other devices. Typically, such a device is powered by an onboard battery or batteries. Such devices require configuration using CHoPAN (Chopan).

At the time of this document, Sedona Framework support for hibernating devices is *not widely available*. However, the SedonaJen6lpNetwork driver in the NiagaraAX station is "ready" for such device support. This is also noted in other sections of this document that reference "hibernating devices".

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.

Note other Jennic-based devices exist that do *not* use the Sedona Framework application layer. Currently no Tridium products or tech docs apply to those "non-Sedona Framework" Jennic-based devices.

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. In a SedonaJen6lpNetwork, the JACE station is always the top "coordinator" node of the network tree.

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 Sedona Framework 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. Sedona Sox Tools in Workbench include a "Kit Manager" view to manage kits on a Sedona device.

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 (or "Sedona device") is Sedona Framework Certified, and runs a Sedona app in a Sedona VM (virtual machine), using installed Sedona kits, and is configured in Workbench using a Sox connection. Devices may vary in a number of ways, including device connectivity—for example Ethernet/IP, WiFi, or 802.15.4 (wireless PAN). Currently, Chopan applies only to wireless Jennic-based devices.

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, and to do initial configuration. A Niagara station also uses Sox to discover, and if so configured, to read and write to Sedona proxy points. However, if a Jennic-based device is configured with Chopan as a Chopan server, 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 *Sedona Framework Chopan Usage* document are listed below.

- Published: November 9, 2011
  Initial document.

CHAPTER 1

Chopan software and hardware requirements

At the time of this document, Chopan support for Sedona Framework devices applies only to wireless (Jennic-based) devices integrated into NiagaraAX, under a SedonaJen6lpNetwork in a JACE station.

Note: *Chopan is not available in Ethernet or WiFi-equipped Sedona Framework devices, which can be integrated into a NiagaraAX station under a SedonaNetwork.*

The following software and hardware items are required:

- NiagaraAX Workbench AX-3.6 or later, enabled for Sedona Framework TXS 1.1 via the Sedona Installer tool. This provides the Workbench engineering environment needed for Chopan.
- A JACE-2, -6, -7 series or JACE-x02 XPR controller (with a wireless Sedona Jennic option card) as the NiagaraAX host platform, running AX-3.6 or later. The JACE acts as the network “coordinator” for the job’s wireless Jennic-based devices. Its station models these devices in the SedonaJen6lpNetwork.
- One or more wireless Jennic-based devices (Sedona Framework device based on a Jennic microcontroller), with the necessary Sedona kits installed in each to support Chopan.

Licensing, module, and kit requirements

Note: *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 TXS 1.1 bundle for Workbench, and restart Workbench. This ensures the latest Sedona-related modules are available in Workbench for installation in remote JACE controllers. For details, see the* NiagaraAX Sedona Installer Guide *document.*

That document also includes a complete summary of Sedona licensing in NiagaraAX.

- The SedonaJen6lpNetwork requires the JACE host to have the *sedonanet*, *nsedona*, *jen6lp*, and *platJen6lp* modules installed, and its license to have the `jen6lp` feature. Device and/or proxy point limits may exist. See the *NiagaraAX Sedona Networks Guide* for more details.
- If the JACE host is to act as a Chopan *server* to networked Jennic-based devices (typical), its `jen6lp` feature must include the attribute entry: `export="true"`
- The license in the JACE host also requires the `tunneling` feature with the attribute `sox="true"`, to allow Sox connections to Jennic-based devices.

Apart from installing the 3*.6.nn* version of the Niagara distribution in the JACE, make sure to install the modules noted above, plus any others needed for other drivers or features. Upgrade any modules shown as “out of date”. For details, see “Software Manager” in the *Platform Guide*.

CHAPTER 2

Sedona Framework Chopan Quick Start

This section provides procedures to start using Chopan in a network of wireless Jennic-based devices. Configuration requires a NiagaraAX Workbench (Fox) connection to the JACE station with an existing SedonaJen6lpNetwork, configured to communicate with child SedonaJen6lpDevices. Some procedures require a (tunneled) Sox connection to one or more of those devices modeled in that network.

Note: *See the "Sedona Network Quick Start" section in the* NiagaraAX Sedona Network Guide *for related details.*

Below are the Chopan-related topics and procedures included in this section:

Setting up Chopan servers

Setting up Jennic-based devices as Chopan servers

Any non-hibernating, networked, Jennic-based device can operate as a CHoPAN server. This lets its app respond to read and write CHoPAN client requests from the JACE, and/or from other networked Jennic-based devices.

Note: *Server functionality for a device includes support for Sedona proxy points, to allow the most efficient tuning policy "Comm Type" (Chopan)—even if not otherwise making Chopan point requests to it.*

- Add chopan kit to Jennic-based device
- Add Chopan server components to App

Add chopan kit to Jennic-based device

With the JACE station open in NiagaraAX Workbench:

Step 1 In the Nav tree, expand the **`SedonaJen6lpNetwork`** to reveal the child **`SedonaJen6lpDevices`**.

Step 2 Right-click a device, and select **`Open Sox - Tunnel Session`**.

An Authentication dialog appears for the Sox connection.

Step 3 Enter the Username and password of a User in the Sedona 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.

Note: *If the connection fails, it may be because your Workbench host does not have the necessary manifest files for all the kits installed in the Sedona 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 users service and your station login. As a workaround, in the JACE station's SoxTunnel service (Config > Services > SoxTunnel), set the "Authenticate with User Service" property to `false` *and* **`Save`**.

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

Step 4 Add the `chopan` kit to the device (if not already present) using the Kit Manager.

To do this, expand the tunneled Sox connection to select **`Tools`** and double-click on the **`Kit Manager`** to access the **`Kit Manager`** wizard. If the `chopan` kit is already present (and up-to-date), skip ahead to another procedure, such as "Add Chopan server components to App".

Otherwise, install/upgrade the `chopan` kit and select **`Restart`** when the wizard completes. This drops the Sox connection, and the connection becomes "ghosted" in the Workbench Nav tree.

Note: *For complete details, see "About the Kit Manager" in the* Sedona Framework Sox Tools Guide.

Step 5 In the Nav tree, double-click the ghosted Sox connection for the device, to reopen it in Workbench.

Add Chopan server components to App

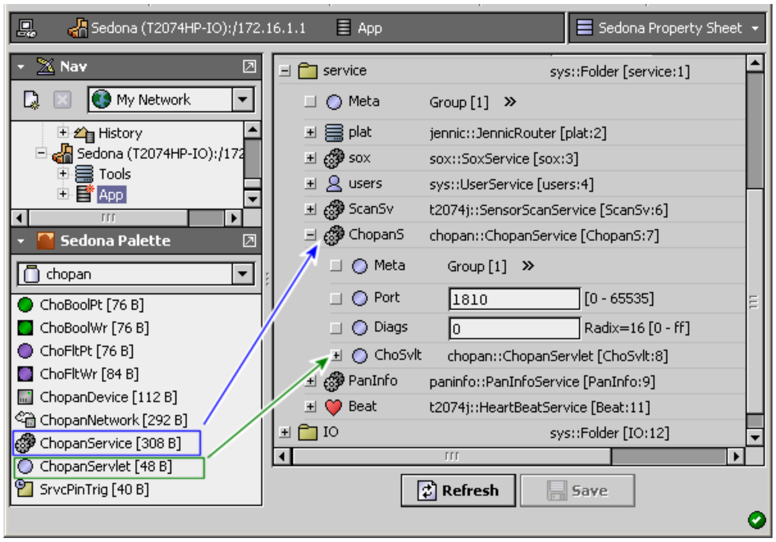
If the Sedona Framework device has the `chopan` kit installed, you can configure it as a Chopan server. See the previous procedure "Add chopan kit to Jennic-based device".

Step 1 Using NiagaraAX Workbench, open a tunneled Sox connection to the device.

Step 2 From the device's **`Nav Container View`**, double-click the **`App`** node, for its **`Sedona Property Sheet`** view, and expand its `service` container (folder).

Step 3 Open the **`Sedona Palette`** in the Workbench sidebar, and choose the `chopan` palette.

Figure 2-1 *Chopan components required for server operation (recommended hierarchy shown)*



service sys::Folder [service:1]
Meta Group [1] »
plat jennic::JennicRouter [plat:2]
sox sox::SoxService [sox:3]
users sys::UserService [users:4]
ScanSv t2074j::SensorScanService [ScanSv:6]
ChopanS chopan::ChopanService [ChopanS:7]
Meta Group [1] »
Port 1810 [0 - 65535]
Diags 0 Radix=16 [0 - ff]
ChoSvlt chopan::ChopanServlet [ChoSvlt:8]
PanInfo paninfo::PanInfoService [PanInfo:9]
Beat t2074j::HeartBeatService [Beat:11]
IO sys::Folder [IO:12]

From the `chopan` palette:

1. Add the **`ChopanService`** to the `service` folder of the App (if the ChopanService is not already present in the App). By default, the component's name is trimmed to "ChopanS".
2. Add the ○ **`ChopanServlet`** under the **`ChopanService`** (again, if not already present). By default, the component's name is trimmed to "ChoSvlt".

Note: *This component hierarchy (App > service > ChopanS > ChoSvlt) is the recommended convention.*

Step 4 If using default Chopan ports (1810), no further configuration is needed.

Note: *If specifying a non-default Port in the property sheet of the* **`ChopanService`**, *remember a matching port number must be used later when configuring any client-side* **`ChopanDevice`** *that represents this device.*

Step 5 Save the App (right-click the **`App`** node and select **`Actions > Save`**).

The device will now support Chopan client requests from remote devices.

Sometimes, you may need to enable a device to make its own Chopan client requests, say if requesting data directly from another Jennic-based device. If so, see "Add core Chopan client components to App".

Setting up NiagaraAX as a Chopan server

Note: *The main use case for the station to operate as a Chopan server is to support client Chopan client requests from networked hibernating devices—which at the time of this document are not widely available. Until full support for hibernating devices is in place, consider this section more "informational" than practical.*

The JACE station running the SedonaJen6lpNetwork can operate as a CHoPAN server, providing it is properly licensed (license feature `jen6lp` must have attribute `export="true"`). This lets the station respond to CHoPAN client read and write requests from networked Jennic-based devices.

- Verify station's Chopan Server
- Understand/set Chopan authorization

Verify station's Chopan Server

With the JACE station open in NiagaraAX Workbench:

Step 1 Open the *property sheet* of the **`SedonaJen6lpNetwork`**, if not already open.

Step 2 Expand the **`Chopan Server`** container to verify the software **`Port`** number (`1810` is default), the server is **`Enabled`** (`true`, also default), and the **`Status`** is `ok`.

You can specify another "non-default" Port number; however, you will need to specify this same port in the client-side **`ChopanDevice`** component that represents the JACE, when configuring the Sedona app in any networked Jennic-based device.

Debug output to the station's Application Director is available, but disabled by default.

Understand/set Chopan authorization

With the JACE station open in NiagaraAX Workbench, after enabling the network’s Chopan Server:

Step 1 In the Nav tree, expand the station’s **`Config`** node to reveal **`Services`** and **`UserService`**, and double-click the **`UserService`** for the **`UserManager`** view.

Find the User named `CHoPAN`. This user is automatically created, with no permissions assigned by default.

Note: *The station automatically allows Sedona reads of any appropriate component, but writes to station components are checked for authorization (as part of this CHoPAN user scheme). This is necessary because CHoPAN is an unauthenticated protocol, meaning that CHoPAN servers do not authenticate or prevent and reads or writes to local components.*

Step 2 As necessary, assign permissions to user CHoPAN, selecting categories and whatever level write (W) privileges may be needed.

Any incoming request that results in a modification of the station database will be checked against the permissions given to this User. In order for the station to accept a write from CHoPAN, the component being written must be assigned to a category for which the CHoPAN User has write privileges.

Note: *If you are familiar with the NiagaraAX BACnet export authentication scheme (via a User named BACnet), please note that this is the identical model.*

For related details, see “Chopan authorization” on page 3-8.

Setting up a Jennic-based device as a Chopan client

Any networked Jennic-based device can operate as a Chopan client. This lets its app initiate read and write client CHoPAN requests directly from the Chopan server in other networked Jennic-based devices, or from the Chopan server of the JACE. If a hibernating device, Chopan *client configuration is required*, but is otherwise optional.

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.*

The device requires the `chopan` kit installed, if not already present. For a quick start procedure, see “Add chopan kit to Jennic-based device” on page 2-2.

These sections apply to Chopan client configuration, and are best understood before beginning setup:

- “Recommended “Best Practices” for NiagaraAX Chopan client integration” on page 2-4
- “Client side Chopan procedures for a Jennic-based device” on page 2-5
  - “Add core Chopan client components to App” on page 2-5
  - “Chopan Device Manager quick start” on page 2-6
  - “Chopan Point Manager quick start” on page 2-7

Recommended “Best Practices” for NiagaraAX Chopan client integration

If making use of the Chopan server in the JACE station, allowing one or more Jennic-based devices to make Chopan read or write client requests to NiagaraAX, some best practices are recommended. These practices include:

1. *Before* adding Chopan points in a device (typically using the “Chopan Virtual gateway” under a SedonaJen6lpDevice in the JACE’s station), copy “ChopanTargetPoints” from the NiagaraAX `jen6lp` palette into a folder you created under that device component. For example, create a folder named “ChopanPts” under the device, at the same level with “Points”, “Parent Pan Info”, and so on.

   Two ChopanTargetPoint types are in `jen6lp` palette: **`ChoBoolTarget`** and **`ChoFltTarget`**. Each is a conveniently “trimmed” standard BooleanWritable or NumericWritable component, with all actions and all inputs but “in10” *hidden*.
   - If reading (pulling) the value from the station, the client request is to the “out” property of this component. The “in10” property is linked in the station to another source component.
   - If writing (pushing) the value from the app, the client request is to the “in10” property of this component. The “out” property is linked in the station to another target component.
2. Copy one ChopanTargetPoint for each data item you are writing (from app to station) or reading (from station to app) into that folder.
3. Rename each ChopanTargetPoint to something descriptive, yet compact—as each will be a “target” for a Chopan point created in the Sedona app, with default names truncated at 7 characters. For example, if adding a ChoFltTarget to hold the out value of a NumericShedule in the station, used for a temperature cooling setpoint, then a name of “ClgSP_S” would help make this clear.

Client side Chopan procedures for a Jennic-based device

The following procedures apply when setting up a Jennic-based device as a Chopan client:

- Add core Chopan client components to App
- Chopan Device Manager quick start
- Chopan Point Manager quick start

Add core Chopan client components to App

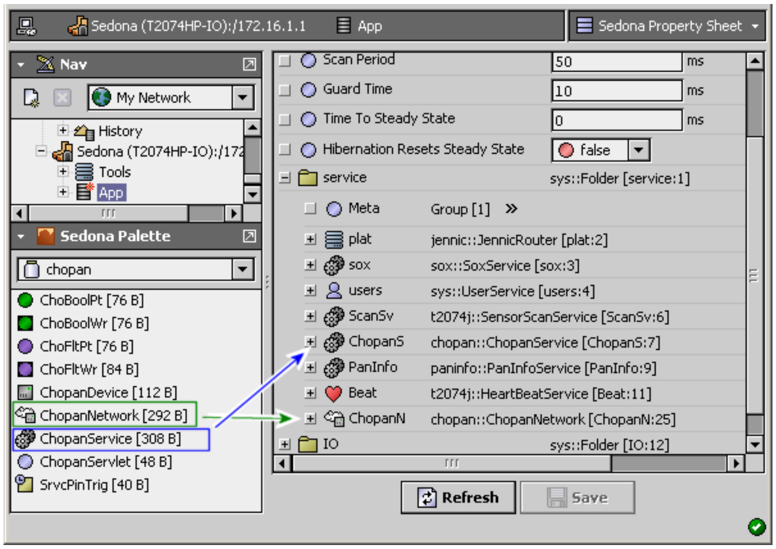
If the Jennic-based device has the `chopan` kit installed, you can configure it with the necessary core Chopan client components.

Step 1 Using NiagaraAX Workbench, open a tunneled Sox connection to the device, if not already open.

Step 2 From the device's **`Nav Container View`**, double-click the **`App`** node, for its **`Sedona Property Sheet`** view, and expand its `service` container (folder).

Step 3 Open the **`Sedona Palette`** in the Workbench sidebar, and choose the `chopan` palette.

Figure 2-2 *Chopan components required for client operation*



Scan Period 50 ms
Guard Time 10 ms
Time To Steady State 0 ms
Hibernation Resets Steady State false
service sys::Folder [service:1]
Meta Group [1]
plat jennic::JennicRouter [plat:2]
sox sox::SoxService [sox:3]
users sys::UserService [users:4]
ScanSv t2074j::SensorScanService [ScanSv:6]
ChopanS chopan::ChopanService [ChopanS:7]
PanInfo paninfo::PanInfoService [PanInfo:9]
Beat t2074j::HeartBeatService [Beat:11]
ChopanN chopan::ChopanNetwork [ChopanN:25]
IO sys::Folder [IO:12]

From the `chopan` palette:

1. Add the **`ChopanService`** to the `service` folder of the app (if the ChopanService is not already present in the app). By default, the component's name is trimmed to "ChopanS".
2. Add the **`ChopanNetwork`** to the `service` folder of the app (if the ChopanNetwork is not already present in the app). By default, the component's name is trimmed to "ChopanN".

   Note: *This component hierarchy (App > service > ChopanS, and App > service > ChopanN is the recommended convention.*

Step 4 Save the app (right-click the **`App`** node and select **`Actions > Save`**).

The device will now support Chopan client configuration from the NiagaraAX station running on the JACE, via the "Chopan Virtual gateway" under its corresponding SedonaJen6lpDevice component.

Step 5 Under the **`ChopanNetwork`**, add one or more **`ChopanDevices`** with child Chopan points to configure data reads and writes.

- The recommended method is to use the views under the "Chopan Virtual gateway" of the **`SedonaJen6lpDevice`** in the JACE's station, meaning the **`Chopan Device Manager`** and then **`Chopan Point Manager`**. Both views provide "Discovery" to simplify the selection of items. This creates the appropriate Sedona components in the app of the represented device, under its **`ChopanNetwork`** component. See "Chopan Device Manager quick start" on page 2-6 and "Chopan Point Manager quick start" on page 2-7.
- Alternatively, it is possible to add **`ChopanDevice`** and Chopan points (**`ChoBoolPt`**, **`ChoBoolWr`**, **`ChoFltPt`**, **`ChoFltWr`**) directly in the device's Sedona Framework app, copying them from the `chopan` Sedona Palette, and locating them under the **`ChopanNetwork`**. However, proper slot addressing (device and URI) is typically difficult.

Chopan Device Manager quick start

Providing the Jennic-based device's app has core Chopan client components (see "Add core Chopan client components to App" on page 2-5), you can use the **Chopan Device Manager** in the device's "Chopan Virtual gateway" to add device-level Chopan client components. This method employs a Sox connection to the device.

With the JACE station open in NiagaraAX Workbench:

Step 1 In the Nav tree, expand the **SedonaJen6lpNetwork** to reveal the child **SedonaJen6lpDevices**.

Step 2 In the Nav tree, expand a device to see its extensions like **Points** and **Chopan Virtual**.

Note: *(Possible future support) If a hibernating device, first invoke the right-click action on the device:* **Maintenance Mode***, and wait for the "mode active" popup in Workbench before doing the rest of this procedure. Be aware that after accessing the Chopan Virtual gateway for a hibernating device, that the device will remain in Maintenance Mode as long as the Nav tree for the Chopan Virtual gateway is open.*

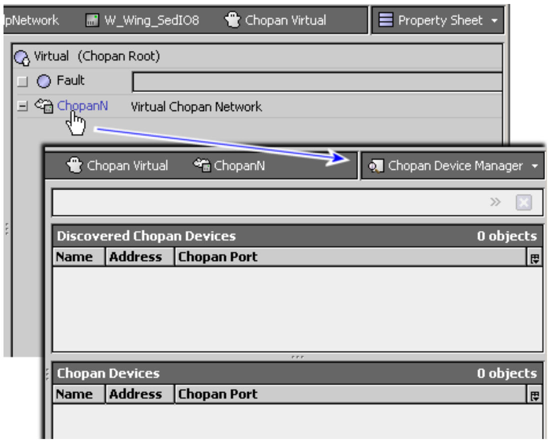
Step 3 Double-click the **Chopan Virtual** gateway.

Its property sheet shows a **ChopanN** (Virtual Chopan Network).

Step 4 Click the **ChopanN** component.

Note: *You must click or double-click components under a Virtual Chopan Network, as opposed to using the expand (⊞) control beside components. This is a "virtual component" characteristic.*

Figure 2-3 *Chopan Device Manager view on the ChopanNetwork virtual component*

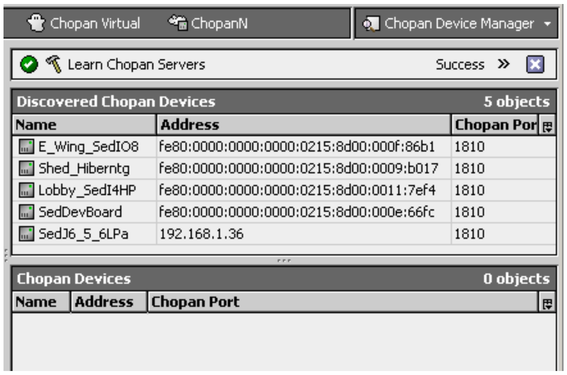


The view changes to the **Chopan Device Manager**, which will initially be blank. Any devices previously added will be listed.

Step 5 Click the **Discover** button.

A "Learn Chopan Servers" job runs, and the view splits in learn mode (top pane lists discovered devices).

Figure 2-4 *Example Chopan Device Manager view after discovery*



Discovered Chopan Devices 5 objects

Name	Address	Chopan Por
E_Wing_SedIO8	fe80:0000:0000:0000:0215:8d00:000f:86b1	1810
Shed_Hiberntg	fe80:0000:0000:0000:0215:8d00:0009:b017	1810
Lobby_SedI4HP	fe80:0000:0000:0000:0215:8d00:0011:7ef4	1810
SedDevBoard	fe80:0000:0000:0000:0215:8d00:000e:66fc	1810
SedJ6_5_6LPa	192.168.1.36	1810

Chopan Devices 0 objects

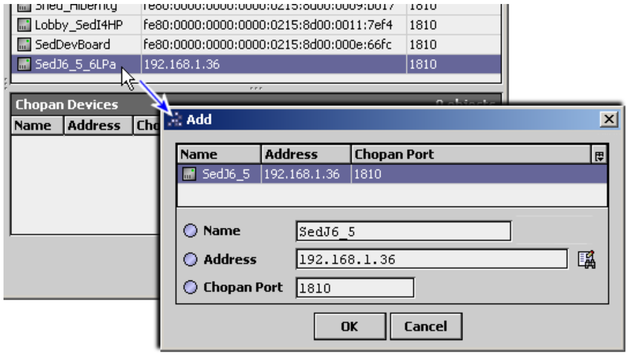
Name	Address	Chopan Port

Discovered Chopan Devices appear as follows:

- Sedona devices are listed by the name of the corresponding SedonaJen6lpDevice in the station, and show their unique 64-bit IPv6 address.
- The JACE (if enabled as a Chopan server) is listed by its station name, and shows its IPv4 address.

Step 6 Double-click a discovered device for an **Add** dialog popup (Figure 2-6).

Figure 2-5 *Example Add dialog when adding a Chopan server*



Lobby_SedI4HP	fe80:0000:0000:0000:0215:8d00:0011:7ef4	1810
SedDevBoard	fe80:0000:0000:0000:0215:8d00:000e:66fc	1810
SedJ6_5_6LPa	192.168.1.36	1810

Name	Address	Chopan Port
SedJ6_5	192.168.1.36	1810

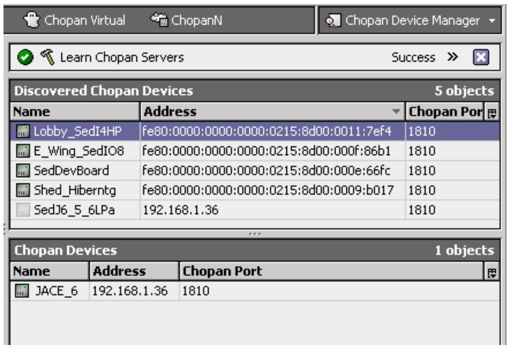
○ Name SedJ6_5
○ Address 192.168.1.36
○ Chopan Port 1810

Add dialog fields described as follows:

- **Name**
  The name of the **`ChopanDevice`** component that will be created in the device's Sedona app. Because Sedona components have a 7-character maximum name, the default name is usually truncated from the original NiagaraAX name. Often, you change this to a more meaningful name.
- **Address**
  IP address of the Chopan server, either IPv6 (Jennic-based device) or IPv4 (JACE). Do not edit.
- **Chopan Port**
  The software port on which this Chopan server listens—typically, the default 1810 is used. However, if the Chopan server is configured to use another port, set this to match.

Step 7 Click **OK** to add it, where it is listed in the "Chopan Devices" lower pane (Figure 2-6).

Figure 2-6 *Example added Chopan server (JACE)*



Discovered Chopan Devices 5 objects

Name	Address	Chopan Por
Lobby_SedI4HP	fe80:0000:0000:0000:0215:8d00:0011:7ef4	1810
E_Wing_SedIO8	fe80:0000:0000:0000:0215:8d00:000f:86b1	1810
SedDevBoard	fe80:0000:0000:0000:0215:8d00:000e:66fc	1810
Shed_Hiberntg	fe80:0000:0000:0000:0215:8d00:0009:b017	1810
SedJ6_5_6LPa	192.168.1.36	1810

Chopan Devices 1 objects

Name	Address	Chopan Port
JACE_6	192.168.1.36	1810

In this example, the name of the device was changed to "JACE_6" before adding.

In the Chopan Virtual gateway device's Sedona app, the corresponding **`ChopanDevice`** component is now under its **`ChopanNetwork`**.

Step 8 Now to access the **`Chopan Point Manager`** for any device, simply double-click the device in the lower pane. See "Chopan Point Manager quick start".

Note: *At the time of this document, there is no "Edit" for a device added in the Chopan Device Manager, although if necessary you can delete a device by selecting it and clicking the Delete (☒) tool on the toolbar above. Otherwise, to edit the component (including its name), open a Sox connection to the device and make changes to it in the device's app.*

Chopan Point Manager quick start

Use the **`Chopan Point Manager`** to select data items to initiate (client) Chopan read or write requests. There must already be at least one Chopan Device under the "Virtual Chopan Network", added using the **`Chopan Device Manager`**. See "Chopan Device Manager quick start" on page 2-6 for related details.

With the JACE station open in NiagaraAX Workbench:

Step 1 If you are not already there, go to the **`Chopan Device Manager`** under the **`Chopan Virtual`** gateway of a **`SedonaJen6lpDevice`** under the **`SedonaJen6lpNetwork`**.

To do this, expand the **`SedonaJen6lpNetwork`** to reveal child **`SedonaJen6lpDevices`**, expand a device and double-click its **`Chopan Virtual`** gateway, and click the **`ChopanN`** (Virtual Chopan Network) for its **`Chopan Device Manager`** view.

Note: *If a hibernating device, first invoke the right-click action on the device:* **`Maintenance Mode`***, and wait for the "mode active" popup in Workbench. Otherwise, you will not see the "ChopanN" node mentioned above, and the Chopan Virtual gateway will remain in fault.*

Step 2 Double-click a device listed under "Chopan Devices".

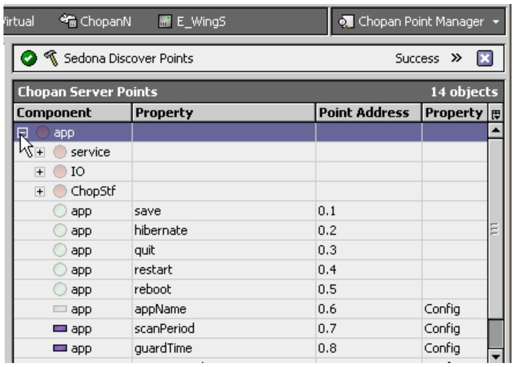
The view changes to the **`Chopan Point Manager`** for that device, which may initially be blank. Any previously defined Chopan points will be listed.

Step 3 Click the **`Discover`** button.

Depending if you selected a Sedona device or the JACE Chopan server, one of two results occurs:

- If a Sedona device, a "Sedona Discover Points" job runs, and the view splits in learn mode (top pane has an expandable app node). See Figure 2-7 below.

Figure 2-7 *Example discovered app node of a Sedona device (shown expanded)*



Sedona Discover Points Success

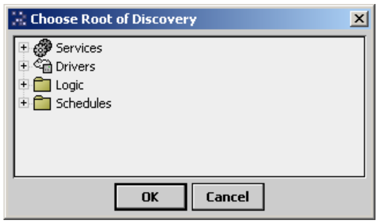
Chopan Server Points 14 objects

Component	Property	Point Address	Property
app			
service			
IO			
ChopStf			
app	save	0.1	
app	hibernate	0.2	
app	quit	0.3	
app	restart	0.4	
app	reboot	0.5	
app	appName	0.6	Config
app	scanPeriod	0.7	Config
app	guardTime	0.8	Config

In this case, skip ahead to Step 5.

- If the JACE Chopan server, a "Choose Root of Discovery" popup dialog appears, showing an expandable tree of containers. See the following Figure 2-8.

Figure 2-8 *Example "Choose Root of Discovery" showing station's Config containers*



Choose Root of Discovery

- Services
- Drivers
- Logic
- Schedules

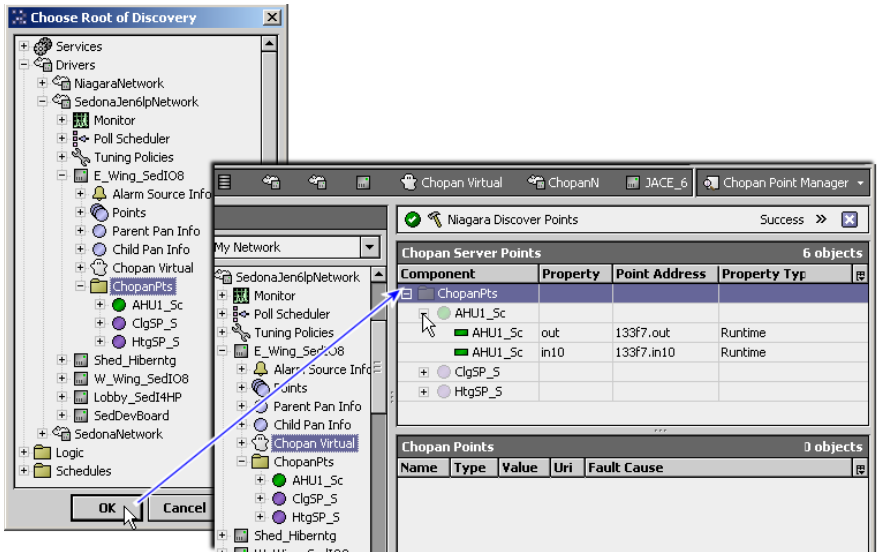
This tree reflects containers in the root of the station's Config node. See Step 4.

Step 4 In the "Choose Root of Discovery" dialog for the JACE station, expand it as needed to find the container holding the component of interest, and select it. Click **`OK`**.

Note: *If using "best practices", this would be a folder created under the SedonaJen6lpDevice being configured, which holds "ChopanTargetPoints" especially made for selection. See "Recommended "Best Practices" for NiagaraAX Chopan client integration" on page 2-4.*

After clicking **`OK`**, a "Niagara Discover Points" job runs, and the view is in learn mode.

Figure 2-9 *Selecting example root of discovery, and resulting discovered node*



Chopan Server Points — 6 objects

Component	Property	Point Address	Property Typ
ChopanPts			
AHU1_Sc			
AHU1_Sc	out	133f7.out	Runtime
AHU1_Sc	in10	133f7.in10	Runtime
ClgSP_S			
HtgSP_S			

Chopan Points — 0 objects

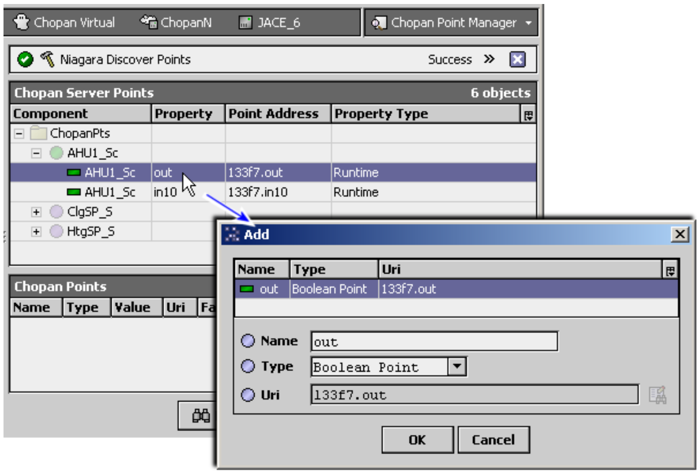
Name	Type	Value	Uri	Fault Cause

The top pane contains the selected folder or container, expandable as shown in Figure 2-9 above.

Step 5 In the top discovered pane (**`Chopan Server Points`**), click to expand the top folder or app node until the property (slot) of interest appears.

Double-click a single row to add one point, or select multiple rows and click ✚ **Add** to add multiple points. A popup **Add** dialog appears (Figure 2-10).

Figure 2-10 *Double-click single property for Add dialog*



Chopan Server Points — 6 objects

Component	Property	Point Address	Property Type
ChopanPts			
AHU1_Sc			
AHU1_Sc	out	133f7.out	Runtime
AHU1_Sc	in10	133f7.in10	Runtime
ClgSP_S			
HtgSP_S			

Chopan Points

Name	Type	Value	Uri	Fa

Add

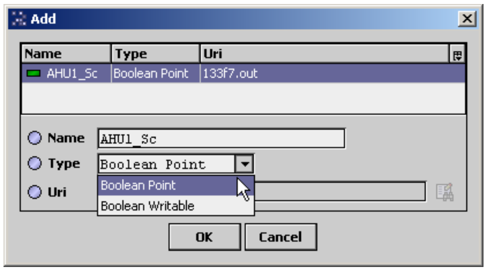
Name	Type	Uri
out	Boolean Point	133f7.out

○ Name: out
○ Type: Boolean Point
○ Uri: 133f7.out

In the Add dialog, note two of the three fields are editable. The three fields are described as follows:

- **Name**
  The default reflects the name (or some portion) of the source property—for example “out” or “nextVal”. As Sedona components have a 7-character name limit, the default name may be *truncated* from the original property name. Typically, you change this to a more meaningful name, working within the 7-character limitation. Figure 2-11 shows a Name after editing in the Add dialog.

Figure 2-11 *Add dialog for Chopan point, where Name has been edited*



Name	Type	Uri
AHU1_Sc	Boolean Point	133f7.out

○ Name: AHU1_Sc
○ Type: Boolean Point (Boolean Point, Boolean Writable)
○ Uri

- **Type**
  Chopan supports two data types: Boolean and Float (numeric), pre-selected according to source property. Each type lets you create a read-only point: “Boolean Point” or “Float Point” (the default), *or* a writable point “Boolean Writable” or “Float Writable”. (Note in the target Sedona app, this creates either a **`ChoBoolPt`**, **`ChoFltPt`**, **`ChoBoolWr`**, or **`ChoFltWr`** component).

  Often, the default read-only point is appropriate—such as when the remote value only needs to be read within a Sedona app. For example, the output of a Schedule component in the JACE’s station.

  In other cases, where the Sedona device’s app needs to write the value, a writable point is needed. For example, the device’s Sedona app needs to write a locally-sourced value to some remote device, say a local temperature value from an I/O component, to be used in the remote device. In this case, you would select a writable Chopan point (Float Writable), so you could link the out of the Sedona temperature component to the writable Chopan point (ChoFltWr). If the JACE station is the remote device, ideally the selected target would be one of the “ChopanTargetPoints” that has already been added to the station (see “Recommended “Best Practices” for NiagaraAX Chopan client integration” on page 2-4).
- **Uri**
  Not editable. Shows the Universal Resource Identifier or “handle” to the target *`component.slot`* in either the JACE station or Sedona device’s app.

Note: *At the time of this document, there is no “Edit” for a point added in the Chopan Point Manager, although if necessary you can* delete *a point by selecting it and clicking the Delete (✕) tool on the toolbar above. Otherwise, to edit the component (including its name), open a Sox connection to the device and make changes to it in the device’s app.*

Step 6 Click **`OK`** to add the point, where it appears listed in the lower “Chopan Points” pane.

In the device’s Sedona app, a corresponding Chopan client point component (**`ChoBoolPt`**, **`ChoFltPt`**, **`ChoBoolWr`**, or **`ChoFltWr`**) is now under the **`ChopanDevice`** that represents the server.

Step 7 When finished adding Chopan points, make a Sox tunnel connection to the device and link these client points (found under `App > service > ChopanN >` *`ChopanDeviceName`*) into other components in its Sedona app, as needed.

For additional **`Chopan Point Manager`** details, see “Chopan Point Manager notes” on page 3-5.

CHAPTER 3

About Chopan in the Sedona Framework

This section provides various concepts and details about CHoPAN, including the applicable components and views in both the JACE's NiagaraAX station as well as Jennic-based devices' apps. These are the main subsections:

Chopan compared to Sox

CHoPAN is a Tridium proprietary protocol running over UDP/IP that enables Sedona Framework devices to share data with JACE stations and other Sedona Framework devices. CHoPAN can be used as an alternative to Sox Polling or Sox Eventing to allow a JACE station to read/write data in Sedona Framework devices.

It is also the only way for Sedona Framework devices to read/write data in other Sedona Framework devices, as there is no Sox client, only a Sox server.

Key comparison points between CHoPAN and Sox:

1. *Sessionless vs. session-based*

   The Sox protocol is designed around an authenticated session between a client and server. CHoPAN is sessionless; each request is independent of the previous one. So the Sox session requires the server to maintain state information about the session, and effectively limits the maximum number of open sessions to whatever resources the server is able to allocate for session management.
2. *Network bandwidth usage*

   As a consequence of point 1, each Sox connection requires multiple messages to set up and tear down and dedicated resources to maintain. There is continuous "keepalive" traffic to maintain the session. CHoPAN is not authenticated, requires no connection setup and no keepalive traffic. This makes CHoPAN a better choice on low-throughput networks like the wireless 802.14.5 Jennic.
3. *Peer-to-peer communication*

   The Sedona Framework does not have a Sox client. So for a Sedona Framework device, the only client option for peer-to-peer communication is CHoPAN.
   Note: *Communication from a Sedona Framework client to a Sedona Framework server via CHoPAN may not be directly peer-to-peer. Due to the JenNet tree structure of a SedonaJen6lpNetwork, a client may need to send its request up one or more nodes in the tree (possibly to the coordinator) to reach the server node.*
4. *Hibernation*

   Any server (Sox or CHoPAN) must be available continuously for client access. So for hibernating nodes, the only logical choice for sharing data is for them to be a CHoPAN client.
   Note: *Currently, Sedona Framework support for hibernating devices (typically battery-powered devices) is not widely available. However, related NiagaraAX support exists in the SedonaJen6lpNetwork driver and NiagaraAX Workbench, and is typically "noted" like this.*

   When the Sedona Framework device contains a CHoPAN client, it can initiate reads and writes from other devices. The device can wake up, use CHoPAN to push and/or pull data from another device, then go back to sleep. Sedona Framework devices cannot use Sox to initiate reads or writes because only the Sox server code is implemented in the Sedona Framework.
5. *Service Pin notification*

   The Sedona Framework device can generate a service pin notification to help user identify physical devices. This is not possible with Sox.

Proxy point usage of Chopan

Perhaps the biggest payback from using Chopan are the efficiencies it provides in polling of Sedona proxy points, providing that Jennic-based devices are configured for Chopan server operation. This applies to any "non-hibernating" device (typically, any device that remains continuously powered).

To take advantage of this, simply configure each Jennic-based device for Chopan server operation, and in the station's proxy points for each device (under the SedonaJen6lpDevice's **`Points`** extension), assign all proxy points to tuning policy(ies) that use "Chopan" as the "Comm Type" selection—the default type.

For more details, see "Setting up Jennic-based devices as Chopan servers" on page 2-2, and various sections in the *NiagaraAX Sedona Framework Networks Guide*, such as "Create Sedona proxy points (and action points", "SedonaJen6lpNetwork properties": tuning policy notes and Chopan comm type considerations.

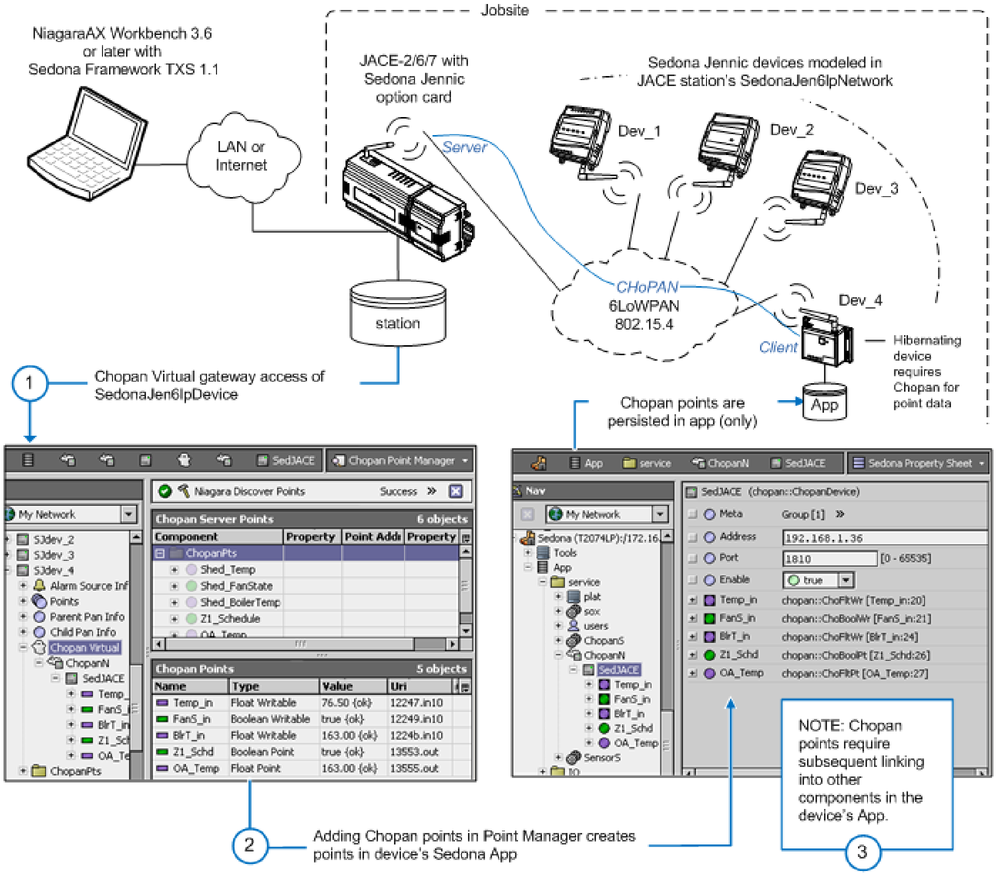
In this scenario, any other client-side usage of Chopan is not typically required—unless perhaps, devices need to share data directly without station (Sedona proxy point) involvement.

Workbench engineering of Chopan client points

You engineer Chopan client points using the “Chopan Virtual gateway” device extension of a selected SedonaJen6lpDevice. Figure 3-1 below shows an example where a hibernating device (Dev_4) has Chopan points added to both read station data and write local app values to the station.

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 support, as described in this example. For another use of Chopan client points that is fully supported now, see the example after this one, “Peer-to-peer configuration example”.*

Figure 3-1 *Access Chopan Virtual gateway using NiagaraAX Workbench to make Chopan (client) points*



Name	Type	Value	Uri
Temp_in	Float Writable	76.50 {ok}	12247.in10
FanS_in	Boolean Writable	true {ok}	12249.in10
BlrT_in	Float Writable	163.00 {ok}	1224b.in10
Z1_Schd	Boolean Point	true {ok}	13553.out
OA_Temp	Float Point	163.00 {ok}	13555.out

SedJACE (chopan::ChopanDevice)

Meta	Group [1] »
Address	192.168.1.36
Port	1810 [0 - 65535]
Enable	true
Temp_in	chopan::ChoFltWr [Temp_in:20]
FanS_in	chopan::ChoBoolWr [FanS_in:21]
BlrT_in	chopan::ChoFltWr [BlrT_in:24]
Z1_Schd	chopan::ChoBoolPt [Z1_Schd:26]
OA_Temp	chopan::ChoFltPt [OA_Temp:27]

NOTE: Chopan points require subsequent linking into other components in the device's App.

In the Figure 3-1 example, once the SedonaJen6lpDevice has been set to “Maintenance Mode”, its Chopan Virtual gateway is accessed in order to add a ChopanDevice representing the JACE server under its ChopanNetwork. Then, as shown on the left side, the **`Chopan Point Manager`** is used to add Chopan points under it—ideally, using target Niagara components already made in a “ChopanPts” folder.

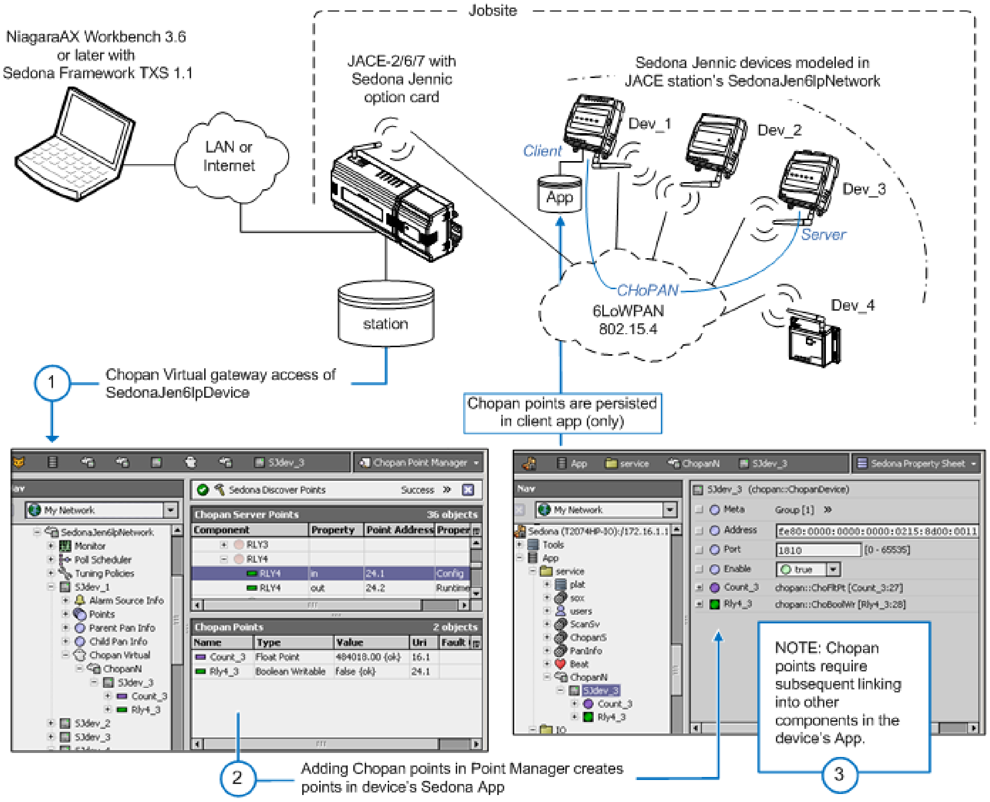
This creates Chopan points in the device’s app (ChoFltWr, ChoBoolWr, ChoBoolPt, ChoFltPt), which are then typically linked into other components in the device’s app (in order to write local values or read/use station values). Chopan points persist only in the Jennic-based device; the Chopan Virtual gateway in the station (used to create them) dynamically reflects corresponding components in the app.

Because of the underlying JenNet tree structure of the wireless network of Jennic-based devices, the communications path of CHoPAN messaging between devices may not be directly between the client and server nodes. For example in the network in Figure 3-1, Dev_4 (as an end node) may be a child node of Dev_2 (acting as a router), such that JACE to Dev_4 messaging involves Dev_2. Note that JenNet uses a “self-healing” tree architecture, such that it automatically reconfigure if routing changes are warranted.

Peer-to-peer configuration example

Chopan points can also allow Jennic-based devices to directly exchange data (peer-to-peer). Engineering is similar to the previous example, beginning with Chopan Virtual gateway access to the client (requesting) SedonaJen6lpDevice in the station. See Figure 3-2.

Figure 3-2 *Chopan Virtual gateway used to make Chopan (client) points in peer-to-peer data exchange*



Component	Property	Point Address	Proper
RLY3			
RLY4			
RLY4	in	24.1	Config
RLY4	out	24.2	Runtime

Name	Type	Value	Uri	Fault
Count_3	Float Point	484018.00 {ok}	16.1	
Rly4_3	Boolean Writable	false {ok}	24.1	

Meta	Group [1]
Address	fe80:0000:0000:0000:0215:8d00:0011
Port	1810 [0 - 65535]
Enable	true
Count_3	chopan::ChoFltPt [Count_3:27]
Rly4_3	chopan::ChoBoolWr [Rly4_3:28]

Chopan Virtual gateway access of SedonaJen6lpDevice

Chopan points are persisted in client app (only)

Adding Chopan points in Point Manager creates points in device's Sedona App

NOTE: Chopan points require subsequent linking into other components in the device's App.

In the Figure 3-2 example, the SedonaJen6lpDevice for Dev_1 has its Chopan Virtual gateway accessed in order to add a ChopanDevice representing Dev_3 under its ChopanNetwork. Then, as shown on the left side, Chopan points are added under that device—in this case one Float Point for a count slot in the Dev_3, and another Boolean Writable for the in slot of a relay 4 output in Dev_3.

This creates Chopan points in the client device's app (ChoFltPt, ChoBoolWr), which are then typically linked into other components in the app (in order to read/use remote count value, write remote relay output). Note that Chopan points persist only in the requesting client Jennic-based device (Dev_1).

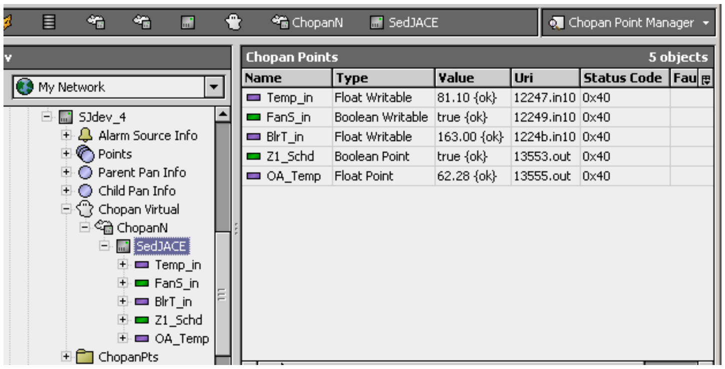
In this case, Dev_1 requires Chopan client configuration, and Dev_3 requires Chopan server configuration. Note that both devices could also be (additionally) configured in a complimentary fashion, say if Dev_3 required client access to read or write values served by Dev_1. However, this is not shown here.

Because of the underlying JenNet tree structure of the wireless network of Jennic-based devices, the communications path of CHoPAN messaging between devices may not be directly between the client and server nodes. For example in the network in Figure 3-2, Dev_3 may be a child node of Dev_2 (acting as a router), such that Dev_1 to Dev_3 messaging involves Dev_2. Note that JenNet uses a "self-healing" tree architecture, such that it automatically re-configures if routing changes are warranted.

Chopan Point Manager notes

The **`Chopan Point Manager`**, the default view on a virtual *ChopanDevice* under the virtual **`ChopanN`**(etwork) of the **`Virtual Chopan`** gateway of a **`SedonaJen6lpDevice`**, provides the means to add and delete client Chopan points in the app of the represented device.

Figure 3-3 *Chopan Point Manager provides selectable columns, including Status Code*



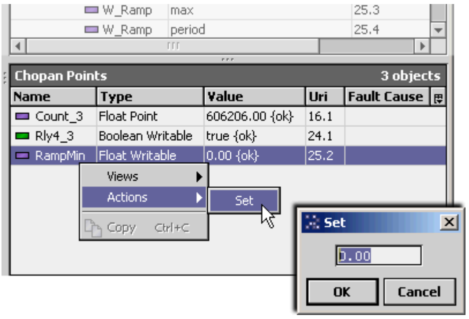
Chopan Points 5 objects

Name	Type	Value	Uri	Status Code	Fau
Temp_in	Float Writable	81.10 {ok}	12247.in10	0x40	
FanS_in	Boolean Writable	true {ok}	12249.in10	0x40	
BlrT_in	Float Writable	163.00 {ok}	1224b.in10	0x40	
Z1_Schd	Boolean Point	true {ok}	13553.out	0x40	
OA_Temp	Float Point	62.28 {ok}	13555.out	0x40	

Additional support is provided in this view (Figure 3-3) for these functions:

- **Diagnostic codes**
  Using the table selector control, you can add “Status Code” to the **`Chopan Points`** pane. This displays a hex numerical code representing the response status of the point’s latest request. A code of `0x40` (decimal 64) means “`OK`”, that is everything is fine.
  Other codes can help diagnose possible issues. Note that in the device’s app, the **`Sedona Property Sheet`** view for any Chopan point provides a “Status Code” property with “decoded” text “Name” string, so you see “`OK`” instead of `0x40`, or “`Unauthorized`” instead of `0x81`. For a listing of possible status code values with names and descriptions, see Table 3-1 on page 10.
- **Manual set** (write).
  For any *writable* point in the Chopan Point Manager, you can right-click it for a **`Set`** (write) action.

Figure 3-4 *Writable points in Chopan Point Manager have available Set action to write*



Chopan Points 3 objects

Name	Type	Value	Uri	Fault Cause
Count_3	Float Point	606206.00 {ok}	16.1	
Rly4_3	Boolean Writable	true {ok}	24.1	
RampMin	Float Writable	0.00 {ok}	25.2	

Set: 0.00

This causes the value to be set on the client, which in turn writes the value to the server.

For step details on using the Chopan Point Manager, see “Chopan Point Manager quick start” on page 2-7.

Data and point types in Chopan

CHoPAN supports two data types in the Sedona Framework client:

- Boolean (two-state) — Chopan Boolean Point or Boolean Writable (ChopBoolPt, ChopBoolWr).
- Float (numeric) — Chopan Float Point or Float Writable (ChopFltPt, ChopFltWr)

When making Chopan client points in the **`Chopan Point Manager`**, discovery makes available only those properties (slots) that are compatible with these two data types.

- If discovering data items in a Sedona Framework device’s *app*, selectable slots for Float points include those with data types `float`, `long`, `double`, `short`, and `int`(eger). Only slots of type `bool` are selectable for Boolean points. Other component slots using other data types, e.g. `Buf` (`string`), as well as actions, are shown “ghosted”—that is they are unavailable.
- If discovering data items in the JACE’s *station*, only components with selectable properties appear—components based on incompatible data types, such as statusString, statusEnum, string, etc., do not

display. Selectable properties for Float points include data types `statusNumeric`, `float`, `double`, `short`, and `integer`. Selectable properties for Boolean points are `statusBoolean` and `boolean`.

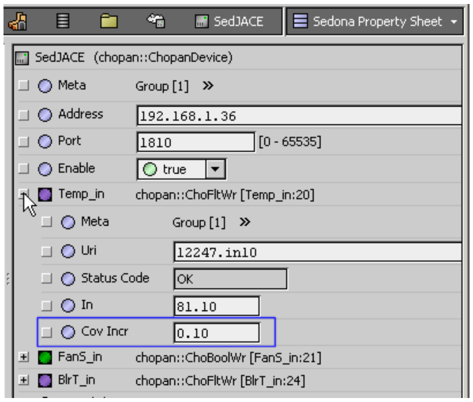
Regardless of target, the "out" slot of a Chopan point in the device's app is either a float (ChoFltPt, ChoFltWr) or a bool (ChoBoolPt, ChoBoolWr).

Note: *The NiagaraAX Chopan client, which is largely Sedona proxy points that use an assigned tuning policy based on the "Chopan" Comm Type, can also process string type data. The JACE platform provides a much larger code space for this type of support than a typical Jennic-based device.*

ChopFltWr COV Increment

Writable Chopan float points (ChoFltWr) have an available "Cov Incr" property (COV increment), as shown on their **`Sedona property sheet`** (Figure 3-5).

Figure 3-5 *Sedona property sheet for ChopFltWr component (writable Chopan float point)*



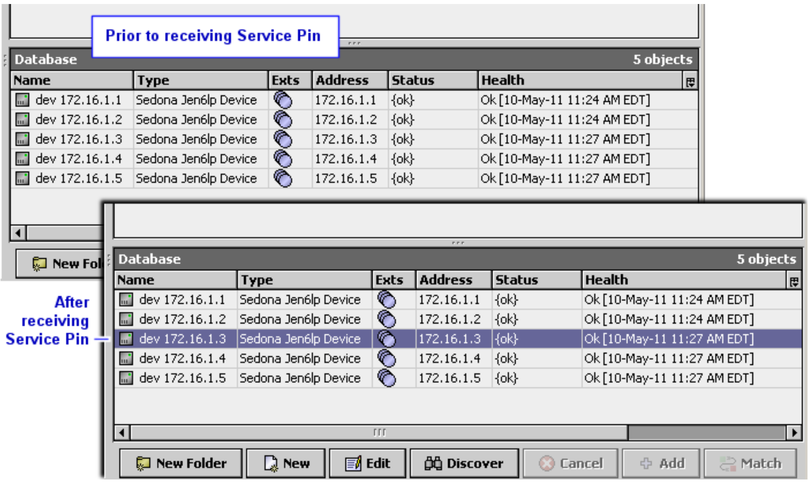
SedJACE (chopan::ChopanDevice)
Meta: Group [1] »
Address: 192.168.1.36
Port: 1810 [0 - 65535]
Enable: true
Temp_in: chopan::ChoFltWr [Temp_in:20]
Meta: Group [1] »
Uri: 12247.in10
Status Code: OK
In: 81.10
Cov Incr: 0.10
FanS_in: chopan::ChoBoolWr [FanS_in:21]
BlrT_in: chopan::ChoFltWr [BlrT_in:24]

You can specify a value larger that the default (0.00) to prevent unnecessary Chopan client requests. This can be useful if the linked input (source) in the device's app has a rapidly fluctuating value.

Service pin support

Chopan provides available support for "service pin" notification from a Jennic-based device, intended as an installation aid. The service pin feature sends a special message from the device to the JACE station, identifying itself. The user interface is in the **`Sedona Device Manager`** view of the station's SedonaJen6lpNetwork, shown in Figure 3-6 below.

Figure 3-6 *Device component becomes "highlighted" if the device's app implements "service pin" routine*



Database 5 objects

Name	Type	Exts	Address	Status	Health
dev 172.16.1.1	Sedona Jen6lp Device		172.16.1.1	{ok}	Ok [10-May-11 11:24 AM EDT]
dev 172.16.1.2	Sedona Jen6lp Device		172.16.1.2	{ok}	Ok [10-May-11 11:24 AM EDT]
dev 172.16.1.3	Sedona Jen6lp Device		172.16.1.3	{ok}	Ok [10-May-11 11:27 AM EDT]
dev 172.16.1.4	Sedona Jen6lp Device		172.16.1.4	{ok}	Ok [10-May-11 11:27 AM EDT]
dev 172.16.1.5	Sedona Jen6lp Device		172.16.1.5	{ok}	Ok [10-May-11 11:27 AM EDT]

Database 5 objects

Name	Type	Exts	Address	Status	Health
dev 172.16.1.1	Sedona Jen6lp Device		172.16.1.1	{ok}	Ok [10-May-11 11:24 AM EDT]
dev 172.16.1.2	Sedona Jen6lp Device		172.16.1.2	{ok}	Ok [10-May-11 11:24 AM EDT]
dev 172.16.1.3	Sedona Jen6lp Device		172.16.1.3	{ok}	Ok [10-May-11 11:27 AM EDT]
dev 172.16.1.4	Sedona Jen6lp Device		172.16.1.4	{ok}	Ok [10-May-11 11:27 AM EDT]
dev 172.16.1.5	Sedona Jen6lp Device		172.16.1.5	{ok}	Ok [10-May-11 11:27 AM EDT]

If a service pin message is received, the device row entry for it in the device manager's database table becomes highlighted, as shown in Figure 3-6. This can be useful after discovering and adding devices, when making positive associations in order to accurately rename device components.

Note: *Only one device row in the Sedona Device Manager view can be highlighted from a service pin message. If service pin messages are received from multiple devices, only the "last one" remains highlighted.*

Service pin setup is described in the following section.

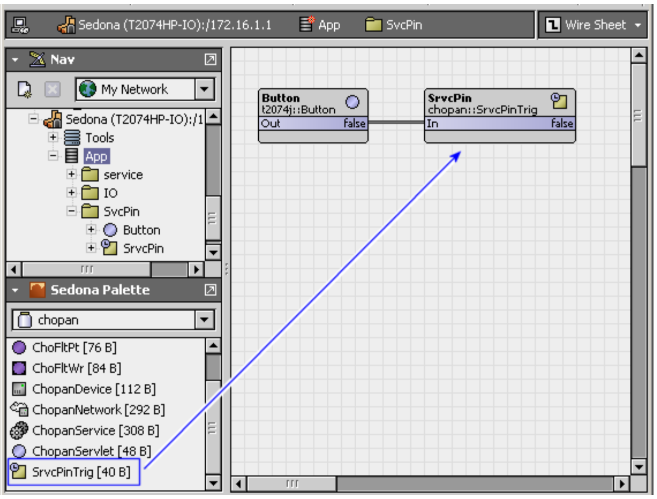
Service pin setup

Support for service pin requires the JACE station has a working SedonaJen6lpNetwork, with communicating SedonaJen6lpDevice child components. Discovery and addition of devices is the typical method. For details, see “Adding discovered SedonaJen6lpDevices” in the *NiagaraAX Sedona Networks Guide*.

The remainder of setup for any Jennic-based device to support service pin messaging is as follows:

- The device needs the chopan kit installed, with its app configured to support Chopan client operation. See “Add chopan kit to Jennic-based device” on page 2-2 and “Add core Chopan client components to App” on page 2-5.
  No further configuration of the ChopanN (ChopanNetwork) component in the app is needed.
- If supported by components in kits available from the device’s vendor, where the device has some local physical input (say, a certain pushbutton switch) that can physically initiate a message, add that component to the device’s app, along with a **`SrvcPinTrig`** component from the `chopan` palette (from **`Sedona Palette`** sidebar of Workbench).
  Such a configuration is shown below, where the two components are in a folder named “SvcPin”.

Figure 3-7 *Example “Button” and SrvcPinTrig components in Jennic-based device’s app*



Link the bool out slot of the initiating component (in this case, Button) to the “in” slot of the SrvcPinTrig component, as shown in Figure 3-7. A transition on the “in” slot of SrvcPinTrig causes the corresponding row for that device in the station’s Sedona Device Manager (if being viewed) to become highlighted.

- Note that another “programmatic” approach is possible, by invoking the “Service Pin” action on the ChopanNetwork component in the device’s app.

Figure 3-8 *Invoking Service Pin action on ChopanNetwork component to trigger service pin message.*

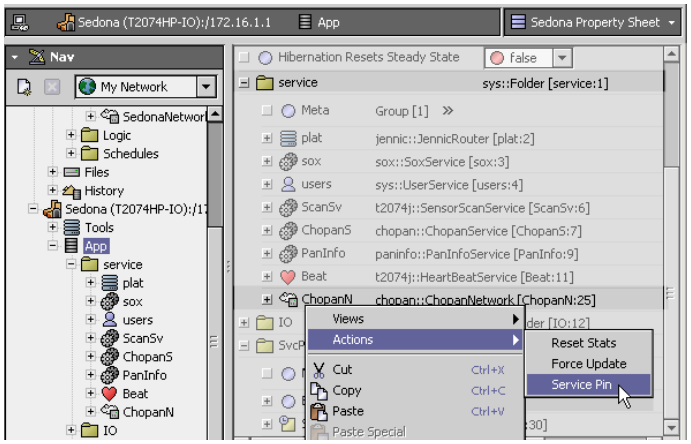


Figure 3-8 shows this action being invoked from the property sheet of a device’s App. This method does not require the **`SrvcPinTrig`** component (from the `chopan` palette) to be used in the app.

Chopan authorization

CHoPAN is an unathenticated protocol. Sedona Framework Chopan servers do not authenticate or prevent any reads or writes to app components. By default, the Chopan server in the Niagara (JACE) station allows *reads* of *any* appropriate component slot. However, *writes* from apps in Sedona Framework devices to components in the station are checked for *authorization*.

This is not part of the CHoPAN protocol, but is internal to the Niagara station. Enabling the Chopan Server in the SedonaJen6lpNetwork automatically creates a User in the station's **`UserService`** named "CHoPAN". See Figure 3-9.

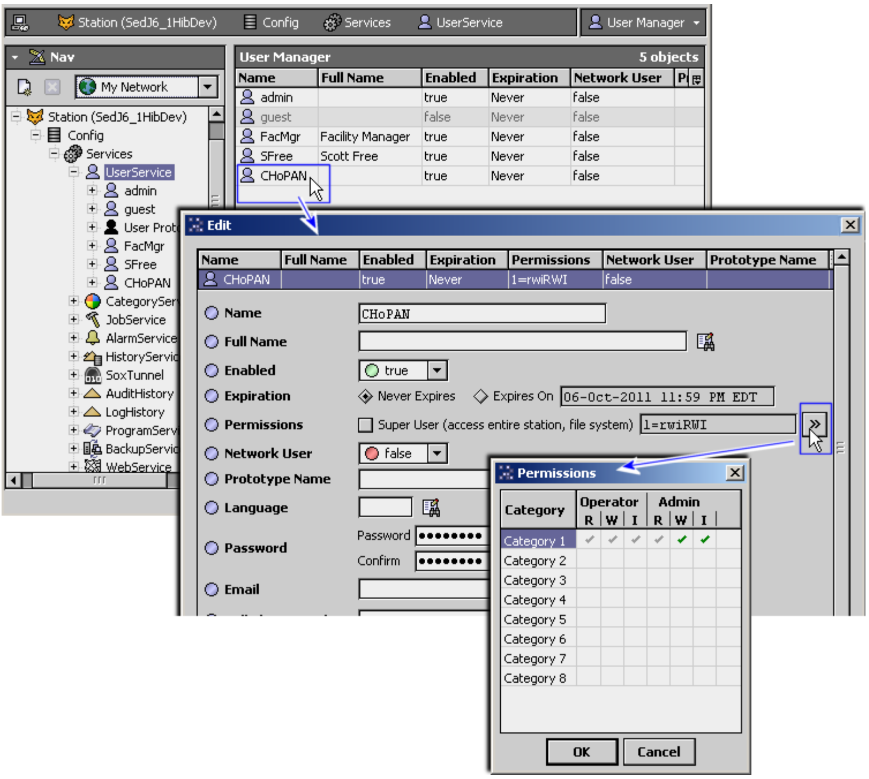


Figure 3-9 *CHoPAN user requires write permissions if writes from app to station are permitted*

User Manager — 5 objects

Name	Full Name	Enabled	Expiration	Network User
admin		true	Never	false
guest		false	Never	false
FacMgr	Facility Manager	true	Never	false
SFree	Scott Free	true	Never	false
CHoPAN		true	Never	false

Edit

Name	Full Name	Enabled	Expiration	Permissions	Network User	Prototype Name
CHoPAN		true	Never	1=rwiRWI	false	

○ Name: CHoPAN
○ Full Name:
○ Enabled: true
○ Expiration: ◈ Never Expires ◇ Expires On 06-Oct-2011 11:59 PM EDT
○ Permissions: ☐ Super User (access entire station, file system) 1=rwiRWI
○ Network User: false
○ Prototype Name:
○ Language:
○ Password: Password •••••••• Confirm ••••••••
○ Email:

Permissions

Category	Operator R	Operator W	Operator I	Admin R	Admin W	Admin I
Category 1	✓	✓	✓	✓	✓	✓
Category 2						
Category 3						
Category 4						
Category 5						
Category 6						
Category 7						
Category 8						

As shown in Figure 3-9, you may wish to set write (W) permissions on one or more categories in the JACE station for user CHoPAN. If using the recommended "best practices", where all components with target properties in the station are under each SedonaJen6lpDevice, in a "ChopanPts" (or similarly-named) folder, then limiting write permissions to the category assigned to the SedonaJen6lpNetwork (with inherited permissions for its child components), would be one logical option.

Maintenance Mode

Note: *At the time of this document, Sedona Framework support for hibernating devices (typically battery powered devices) is not widely available, including "Maintenance Mode" as described here. However, the SedonaJen6lpNetwork driver in the NiagaraAX station is "ready" for such device support.*

Maintenance Mode provides the means for a hibernating Jennic-based device to become "awake" to accept an incoming Sox connection from NiagaraAX Workbench. Typically, this applies only to a battery-powered device, which spends most of the time in a very low-power (hibernation) state, with its Sedona Framework VM and RF communications disabled.

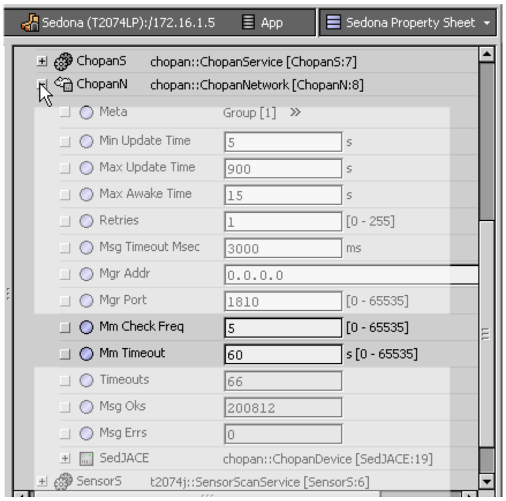
The Workbench user invokes a **`Request Maintenance Mode`** action on the station's SedonaJen6lpDevice that represents the device. This sets a flag in the SedonaJen6lpNetwork's Chopan Server that maintenance mode has been requested.

Note: *For a description of how this works from the NiagaraAX (station) side using Workbench, see "SedonaJen6lpDevice actions" in the* NiagaraAX Sedona Networks Guide.

The Chopan client operating in the device's app makes a periodic Chopan client request, checking this maintenance mode (Mm) flag. If detected set, the device awakens into "Maintenance Mode", allowing a Sox connection to it to be made. Once a Sox connection is made, the device remains awake for the duration of that connection, returning to hibernation only after the Sox session has ended.

Related properties are in the ChopanNetwork component in the device's app. See Figure 3-9.

Figure 3-10 *ChopanNetwork expanded in device's App property sheet, maintenance mode properties*



ChopanS	chopan::ChopanService [ChopanS:7]	
ChopanN	chopan::ChopanNetwork [ChopanN:8]	
Meta	Group [1]	
Min Update Time	5	s
Max Update Time	900	s
Max Awake Time	15	s
Retries	1	[0 - 255]
Msg Timeout Msec	3000	ms
Mgr Addr	0.0.0.0	
Mgr Port	1810	[0 - 65535]
Mm Check Freq	5	[0 - 65535]
Mm Timeout	60	s [0 - 65535]
Timeouts	66	
Msg Oks	200812	
Msg Errs	0	
SedJACE	chopan::ChopanDevice [SedJACE:19]	
SensorS	t2074j::SensorScanService [SensorS:6]	

- Mm Check Freq — specifies the number of hibernation cycles between checks for maintenance mode.
- Mm Timeout — specifies how long, once the device receives the maintenance mode request and becomes awake, that it waits for a Sox connection to be made. The default is 60 seconds.
  This timeout prevents battery drain if a user places a device in maintenance mode, but then forgets to make a Sox connection. It the timeout expires without a connection made, hibernation resumes.

Note: *Any hibernating-type device requires similar configuration, meaning it should have the* `chopan` *kit installed, and be configured as a Chopan client. See the vendor's documentation for any further details.*

Chopan Diagnostics

Currently CHoPAN client points can be either Boolean or numeric, with read-only and writable versions of each. Points in the network are updated in a "round-robin" fashion by walking through the Chopan-Network's child devices, and through each device's child points.

- Read-only points (**`ChoBoolPt`** and **`ChoFltPt`**) are updated by issuing a CHoPAN GET request for their configured URI, and updating the *out* slot with the response value.
- Writable points (**`ChoBoolWr`** and **`ChoFltwr`**) are updated by issuing a CHoPAN PUT request to their configured URI, with the value contained in their *in* slot.
- Each point has a *Status Code* slot that reflects the response status of the latest request. Status codes that may appear are shown below. The friendly text "Name" appears in the **`Sedona Property Sheet`** view for any Chopan point. In the **`Chopan Point Manager`**, the hex numerical 0x*NN* status codes display (see Figure 3-3 on page 5). Table 3-1 provides a listing of possible codes.

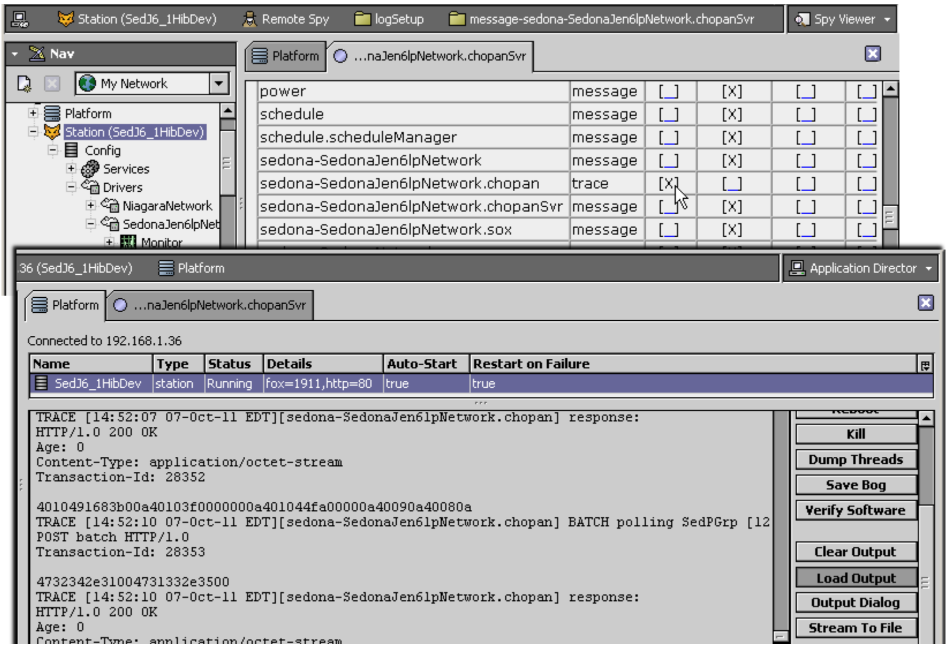
Table 3-1 *CHoPAN Status Codes*

Name	CHoPAN Code	HTTP Code	Description
OK	0x40 (64)	200	Everything is fine—successful read or write.
- -	0x00 (0)	n/a	Point has not been read or written yet.
Decoding Error	0x01 (1)	n/a	There is an error in the decoding value from the sever.
CHoPAN Timeout	0x02 (2)	n/a	No response was received from the server.
Bad Request	0x80 (128)	400	The server rejected the request—see station output for details.
Unauthorized	0x81 (129)	401	The CHoPAN User in the station does not have permissions to modify this component/slot.
Not Found	0x84 (132)	404	The specified URI does not map to a valid component/slot reference.
Server Error	0xA0 (160)	500	The server experienced an internal error—see station output (if JACE server) or serial port diags (if Jennic-based device server) for details.

NiagaraAX Diagnostic output

The (JACE) station running the **`SedonaJen6lpNetwork`** provides a "spy" logSetup page, where you can select processes for trace (verbose) output in the standard output of the station, viewable in the platform's **`Application Director`** view. Among these are two for Chopan messaging (Figure 3-11).

Figure 3-11 *Spy logSetup for station has two Chopan-related process choices*



power	message	[_]	[X]	[_]	[_]
schedule	message	[_]	[X]	[_]	[_]
schedule.scheduleManager	message	[_]	[X]	[_]	[_]
sedona-SedonaJen6lpNetwork	message	[_]	[X]	[_]	[_]
sedona-SedonaJen6lpNetwork.chopan	trace	[X]	[_]	[_]	[_]
sedona-SedonaJen6lpNetwork.chopanSvr	message	[_]	[X]	[_]	[_]
sedona-SedonaJen6lpNetwork.sox	message	[_]	[X]	[_]	[_]

Connected to 192.168.1.36

Name	Type	Status	Details	Auto-Start	Restart on Failure
SedJ6_1HibDev	station	Running	fox=1911,http=80	true	true

```
TRACE [14:52:07 07-Oct-11 EDT][sedona-SedonaJen6lpNetwork.chopan] response:
HTTP/1.0 200 OK
Age: 0
Content-Type: application/octet-stream
Transaction-Id: 28352

4010491683b00a40103f0000000a401044fa00000a40090a40080a
TRACE [14:52:10 07-Oct-11 EDT][sedona-SedonaJen6lpNetwork.chopan] BATCH polling SedPGrp [12
POST batch HTTP/1.0
Transaction-Id: 28353

4732342e31004731332e3500
TRACE [14:52:10 07-Oct-11 EDT][sedona-SedonaJen6lpNetwork.chopan] response:
HTTP/1.0 200 OK
Age: 0
Content-Type: application/octet-stream
```

- **`sedona-SedonaJen6lpNetwork.chopan`** — Reflects Chopan client activity on the station, for example "batch polling" of Sedona proxy points that use a tuning policy with Chopan commType.
- **`sedona-SedonaJen6lpNetwork.chopanSvr`** — Reflects Chopan server activity on the station, in response to incoming Chopan point requests.

Note: *Setting log options to "trace" is meant for temporary diagnostic activity—be sure to return all station log levels to the default "message" setting for normal operation.*

Sedona Framework Diagnostic output

The ChopanService provides a diagnostic bit field to enable different levels of debug output, directed to a local *serial port* on the Jennic-based device. This is typically a "developer-only" level of diagnostics.

This setting is determined by the "Diags" property value of the ChopanService in the device's app, as a hexadecimal value from 0 to ff, where bit 0 (LSB) to bit 7 (MSB) are defined as follows:

- Bit 0 - I/O debug - Traffic sent/received on the ChopanService UDP port (client *and* server).
- Bit 1 - Server debug - Description of requests received by the server (if installed) and actions taken.
- Bit 2 - Client debug - Description of requests issued by the client (if installed).
- Bit 3 - Dot debug - One-character "mini-diags" to indicate state during work cycles.
- Bits 4 through Bit 7 - Unused/undefined.