Technical Document

Sedona Framework TXS-1.2
Offline Engineering Guide

February 14, 2014

Sedona Framework TXS-1.2 Offline Engineering Guide

Confidentiality Notice

Trademark Notice

Copyright and Patent Notice

Contents

Preface

NiagaraAX Workbench 3.7 or later enabled for Sedona TXS-1.2, and Sedona Framework TXS-1.2 Workbench, provide offline engineering features that allow you to create new apps, run apps on simulated devices and to view and modify apps, all while working in offline mode. If using Niagara Workbench, you can also perform offline point discovery and engineer SedonaNetworks without having to be directly connected to the actual hardware.

The following topics covered in this section:

Sedona Framework terms

The following is a list of terms and abbreviations used in this document when describing the Sedona Device Simulator. Complete Sedona Framework documentation is available at the *Sedona Framework website* (http://www.sedonadev.org). For general NiagaraAX terms, see the Glossary in the *Niagara AX User Guide*.

app For application. The app in a Sedona Framework device is its collection of Sedona Framework components, including links between them, plus configuration properties.

app file association The act of matching a Sedona device's app file to one of the Sedona devices residing under a SedonaNetwork in the station database on the JACE is referred to as *association.* App files reside in the JACE file system.

bootstrapping A common software development term, it refers to loading an application using a much smaller initial app to initiate and load more complex processes. As used in this document, "bootstrapping" describes the process of loading the simulator SVM with a default scode image and default app file for the platform. Once the SVM is bootstrapped with the default app, the standard Sedona Framework TXS provisioning tools are used to connect to the SVM and re-provision it with the app you have selected to run in the simulator.

kit Sedona Framework kits are the basic unit of modularity of Sedona Framework 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 to change a device's "core" software. Sedona Framework Sox Tools in Workbench include a "Kit Manager" view to manage kits on a Sedona Framework device.

manifest Each kit has a corresponding manifest, that contains all the metadata needed by tools like Workbench to connect via Sox to a Sedona Framework device, and for a station to support Sedona Framework 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. Use the Manifest Manager view of either Sedona Framework network type to manage kit manifests on the NiagaraAX host (JACE or Supervisor). Use the Sedona Manifest Manager view in Workbench tools to manage manifests on your Workbench host. For more details see the *Sedona Manifest Manager* document.

PAR For Platform Archive, or PAR file. This is a zip file with a .par extension that provides a way of organizing various files and metadata about a platform into a single entity. The contents of the platform archive includes an XML file that describes the platform.

sandbox The Simulator runs in a "sandbox" folder called "`sedonaDeviceSim`" which is created under your user folder: `<niagara-release>/users/<username>/SedonaDeviceSim`). The SVM, scode, and app files are copied to this folder and run from this location. Also, the provisioning process writes files to this location.

SAB For Sedona App Binary. This is a compact binary representation of a Sedona Framework app suitable for storage and execution on a Sedona Framework device. One of two file formats (.sab) for a Sedona Framework app.

SAX For Sedona App XML. This is a simple XML representation of a Sedona Framework app that is easily generated and consumed by Sedona Framework software tools. One of two file formats (.sax) for a Sedona Framework app.

scode A compiled set of kits in a Sedona Framework code image file with an .scode extension. The scode image is executed by the SVM in the device. The Kit Manager tool is used to modify kits.scode on a Sedona Framework device.

SEDONA HOME The term SEDONA HOME is a system "variable" for the installation-specific directory of your Workbench computer's "root" folder for Sedona Framework files used in Sox connections from Workbench. Beginning with Sedona TXS-1.2, the default location for SEDONA HOME is !/sedona.

simulator app The app file that is loaded and successfully running in the Sedona Device Simulator is the *simulator app*.

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 open a connection to a Sedona Framework device. A Niagara station also uses Sox to discover as well as read and write to Sedona Framework proxy points.

SVM The SVM (Sedona Virtual Machine or VM) executes the Sedona app on a device using the kits installed on that device.

Frequently asked questions

The following frequently asked questions (FAQs) about the Sedona Framework TXS-1.2 offline engineering features are covered in this section:

Sedona offline engineering FAQs

Q: **What is meant by the term, "offline engineering"?**

A: Offline Engineering is a term given to the engineering of Sedona Framework-enabled device applications and Niagara Framework station point discovery without the need for a live connection to the respective Sedona or Niagara device.

Q: **Is there any difference between the Offline Engineering tools available using Sedona TXS and Sedona Workbench?**

A: Yes, Sedona Workbench doesn't support the development of Niagara stations. This means that a Sedona Workbench user cannot discover Niagara station points from an offline Sedona application file using Sedona Workbench.

Sedona New App wizard FAQs

Q: **Which files do I need to have installed on my Workbench PC in order to engineer an offline application file for my target platform?**

A: You need to install the kit files for any components you want to add into the Sedona app, including whatever platform you wanted to create it for (there is a platform service required in every app, which is a component that is contained in a kit file). You also need to install the sedona, sedonac, and nsedona modules to create and manipulate the app in Workbench.

Q: **If I don't have all of the required kits where can I obtain them?**

A: Core sedona kits can be obtained from http://sedonadev.org. Vendor-specific kits may be obtained from the device manufacturer.

Q: **Why can't I create an app for platform xyz?**

A: If you are having a problem creating an app for a specific platform, check to be sure that you have the necessary kits that define that platform in your Sedona installation.

Q: **Is there any difference between the offline Sedona-New App Wizard that is available in Sedona Framework TXS-enabled Niagara Workbench and the offline Sedona-New App Wizard that is available in Sedona Workbench?**

A: No, there is no difference.

Sedona Device Simulator FAQs

Q: **Is there any difference between the Sedona Device Simulator that is available in Sedona Framework TXS-enabled Niagara Workbench and Sedona Device Simulator available in Sedona Workbench?**

A: There is no difference. The tool functions the same in both versions of Workbench.

Q: **What is a device simulator PAR file?**

A: A "device simulator PAR" file is a vendor-supplied platform archive file, that contains all of the files needed to get started with the Sedona Device Simulator. Specifically, the zip file contains a simulator SVM with a default scode image and default app file.

Q: **What is the difference between a PAR file and a "device simulator PAR" file?**

A: A standard PAR (Platform Archive) is a zip file with a .par extension which provides a way of organizing various files and metadata about a platform into a single entity. By comparison, a "device simulator PAR" includes a simulator SVM (Sedona Virtual Machine) which provides a stubbed implementation of any native methods that the Sedona kits require. For more information, see documentation on "Platform Archive" at *http://sedonadev.org*.

Q: **How can I obtain a device simulator PAR file?**

A: If you do not have the device simulator PAR (a standard PAR that also contains a simulator SVM, default scode image, and default app file) for a particular platform, contact the vendor of that Sedona platform. Otherwise, a sample device simulator platform is included in the Sedona Framework TXS-1.2 Bundle. The sample device simulator file is an example, it is not a complete alternative to a vendor supplied PAR file. For more information, see "Sample device simulator platform included in TXS bundle" on page 1-2.

Q: **How do I install a device simulator PAR file and use it with my app?**

A: First, you must have Sedona Framework TXS-1.2 installed. Then, use the Sedona Installer Workbench tool to "Import Sedona environment files". This option allows you to import individual kit, manifest, or platform archive files, as well as zip files which are imported and extracted. For more information, see "Installing a device simulator PAR" on page 1-3.

Q: **Once the app is running in the device simulator, which Sedona Tools are available?**

A: All of the provisioning tools are available for use.

Q: **Can I re-provision a simulator SVM with the Sedona tools?**

A: Yes, you can interact with it as with any other SVM running on a real device.

Q: **Can I simulate the I/O that the actual device would have?**

A: The behavior to simulate the actual hardware is implemented by the simulator SVM supplied by the vendor. So, you will not be able to change that behavior. The SVM provides a stubbed version of any methods implemented in native code, such as I/O servicing, hardware clocks, etc. You could check with the vendor to find out if they offer a simulator that provides additional behaviors, such as providing canned or calculated data.

Q: **What is the effect of "kill" versus "stop" on the saved version of the app?**

A: Both options stop running the SVM. *Kill* saves the app only in a "sandbox" location under your user folder. The app in the sandbox location is overwritten at the start of every **`Run`** process. Stopping the SVM with the kill command does not affect the app in the original location. *Stop* saves the app from the sandbox location, allowing you to choose where to save the file, and you can modify the file name. For more details, see "About the Device Simulator view" on page 2-4

Q: **Can the app running in the device simulator adversely affect my Niagara installation?**

A: No, the app functions will not affect your Niagara installation. The Simulator is run in a "sandbox" folder called "`sedonaDeviceSim`" which is created under your user folder: `<niagara-release>/users/<username>/SedonaDeviceSim`). The SVM, scode, and app files are copied to this folder and run from this location. Also, the provisioning process writes files to this location.

Sedona offline point discovery FAQs

Q: **When creating proxy points, can I drag points from the palette and use the "Matching" feature in the Point Manager?**

A: No, currently points are not available in the palette. The recommended method for creating proxy points in the Point Manager is to right-click a point in the discovered pane and click **`Add`**. "Adding" configures the proxy point with the correct `compId:slotId` address value.

Q: **What file types can point discovery use?**

A: Acceptable file types are either SAX or SAB files.

Q: **What does "offline" mean and can I do "offline" point discovery without running the station on my laptop?**

A: The term "offline" typically means, without a live connection to the device. "Offline" point discovery can also be done with a running station connected to the Sedona device, however, it is actually *file-based* discovery because the discovery entries are generated from the associated app file.

About this guide

This documents how to use the offline engineering features in Sedona Framework TXS-1.2.

Note: *Regarding the Sedona Device Simulator tool, this document covers how to use the Device Simulator tool. It does NOT describe how to generate a Device Simulator SVM for a particular platform. If you do not have the Device Simulator PAR file for a platform, contact the vendor of that Sedona platform.*

Audience

This technical guide is intended for NiagaraAX-trained systems integrators who are familiar with the Sedona environment and for Sedona application developers.

Related documentation

NiagaraAX Sedona Framework TXS-1.2 Networks Guide

NiagaraAX Sedona Framework TXS-1.2 Tools Guide

NiagaraAX Sedona Installer Guide

Sedona Manifest Manager Engineering Notes

Sedonadev.org

Document change log

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

- Revised: February 14, 2014
  A note has been added at the beginning of the section on "Requirements" on page 1-1 explaining the TXS version requirement when installing TXS-1.2 in NiagaraAX-3.8. Additional details are available in the *NiagaraAX Sedona Installer Guide*. Another note was added after the first paragraph in the section on the "About the Sedona - New App tool" on page 2-1. Also, in the Troubleshooting appendix, added the section: "Admin password and admin password confirmation do not match" on page A-1.
- Publication: February 11, 2013
  Initial document for Sedona TXS 1.2. Describes offline engineering features new in Sedona TXS-1.2, the typical offline engineering process workflow, includes content from the preliminary draft for Beta version of the Device Simulator Guide, and a troubleshooting appendix.
- Preliminary Draft for Beta (for the Device Simulator Guide): November 1, 2012
- Initial draft: October 14, 2012

Chapter 1

Getting started

Quick start

The high-level overview shown here demonstrates the basic workflow process for offline engineering:

- **Offline app development**
  Using either NiagaraAX Workbench 3.7 (or later) enabled for Sedona Framework TXS-1.2 or Sedona Framework Workbench 1.2
  1. Creating a new app in offline mode
  2. Configuring components in an offline app
  3. Running an app on a simulated device
  4. Connecting to the app running on a simulated device
  5. Modifying the app running on a simulated device
  6. Making provisioning changes to the app running on a simulated device
  7. Saving the modified simulator app file
- **Offline network engineering**
  Using NiagaraAX Workbench 3.7 (or later) enabled for Sedona Framework TXS-1.2
  8. Discovering the simulated device and adding it to a locally running station
  9. Performing point discovery on the simulated device
  10. Viewing proxy points to confirm expected behavior
  11. Accessing the simulator app via the Sox Gateway
  12. Modifying the simulator app, saving changes and updating station file space
- **Engineered station installation**
  Using NiagaraAX Workbench 3.7 (or later) enabled for Sedona Framework TXS-1.2
  13. Transferring the station to the remote JACE
  14. Installing the vendor-specific files on the JACE
  15. Performing device discovery, match, and associate
  16. Pushing the offline app down to a physical device
  17. Viewing points to confirm behavior

Requirements

Note: *If using NiagaraAX-3.8, and planning to install Sedona Framework TXS, you must use an "update 1" bundle of TXS-1.2: version 1.2.1xx (for example 1.2.100). This bundle is specific for use with AX-3.8, and is not backward compatible with AX-3.7. However if using AX-3.7/AX-3.7u1, use an earlier TXS-1.2 bundle: version 1.2.2x (for example 1.2.28.4). And in the case of AX-3.7u1 (3.7.106 or later), after installing the 1.2.28.4 bundle you need to download and install a patched* `nsedona` *module, version 1.2.28.1.*

For more details, see the latest NiagaraAX Sedona Framework TXS-1.2 Installer Guide.

For NiagaraAX Workbench

If using Niagara Workbench, you need to have the following software installed on your PC:

- NiagaraAX 3.7 (or later) platform
- Sedona Framework TXS-1.2 (or later) Bundle
- Required license features: workbench, sedonanet (or jen6lp), sedonaProvisioning, and sox
- A device simulator PAR file for the hardware device that your apps will run on

For Sedona Framework Workbench

If using Sedona Framework Workbench, you need to have the following software installed on your PC:

- Sedona Framework TXS-1.2 (or later)
- Required license features: workbench, sedona
- A device simulator PAR file for the hardware device that your apps will run on

Note: *A sample device simulator PAR file is included in the Sedona Framework TXS-1.2 Bundle. For details, see Sample device simulator platform included in TXS bundle. Once you have installed the sample file, you can use the Device Simulator tool as demonstrated in this guide.*

About device simulator PAR files

A standard PAR (Platform Archive) is a zip file with a .par extension which provides a way of organizing various files and metadata about a platform into a single entity. By comparison, a "device simulator PAR" includes a simulator SVM (Sedona Virtual Machine) which provides a stubbed implementation of any native methods that the Sedona kits require.

A device simulator PAR is required in order to simulate a device. Typically, the device simulator PAR file is supplied by the vendor of the device you are using.

Note: *If you do not have the device simulator for a particular platform, contact the vendor of that Sedona platform. Otherwise, you can install a sample device simulator platform which is included in the Sedona TXS-1.2 Bundle.*

The following topics are covered in this section:

Sample device simulator platform included in TXS bundle

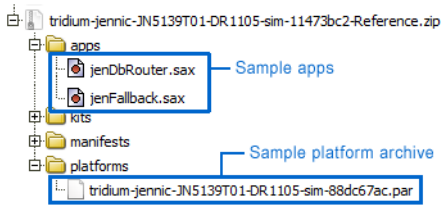
Included in the officially released Sedona Framework TXS-1.2 Bundle is a zip file archive of a sample device simulator platform. The zip file archive installs with this file path: `docs/docSedonaOfflineEng/sampleSimulator/JN5139T01-DR1105-sim-88dc67ac-Reference.zip`. For details on installing the zip file archive, see About installing device simulator platform archives.

Note: *Use of this zip file archive is subject to the accompanying EULA which can be found at this location: docs/docSedonaOfflineEng/sampleSimulator/sampleSimulator-EULA.txt.*

The archive file is in compressed zip file format. The zip file contains the following:

- Two sample app files.
- The kits & manifest files required for the sample apps.
- A simulator PAR based on the Jennic DR1105 device.

Figure 1-1 *Contents of sample device simulator zip*



Limitations engineering Jen6Lp networks using the simulator

There are known limitations to using the sample device simulator platform to engineer a SedonaJen6LpNetwork offline.

When using the Device Simulator, you must use a SedonaNetwork and a SedonaDevice to connect to it, even if the simulator is emulating a Jennic device. However, when you connect to an actual Jennic device, you need to create a SedonaJen6lpNetwork, and add a new SedonaJen6lpDevice to map it. So the work you do simulating the SedonaDevice has limited benefit since you cannot change the Type of a device or network (without editing the XML bog file). Some known issues are:

- On an actual job site, a SedonaJen6lpNetwork must be used to connect to Jen6lp devices. The station used for simulation (which contains a SedonaNetwork) cannot be used on the job site as is.
  - For Jennic-based solutions you need to essentially recreate the network and device, before you can copy the proxy points that you engineered offline.
- For the reasons noted above, the workflow presented in this guide is not recommended for Jennic-based devices.

About installing device simulator platform archives

Install device simulator platform archives using the Sedona Installer tool in Workbench, making sure to select the **`Import Sedona environment files`** option rather than "TXS Bundle". Once you have installed a device simulator platform archive, you can begin using the Device Simulator tool.

Note: *If you are using a Niagara installation of Workbench that does not include Sedona, you will have to install Sedona Framework TXS-1.2 first before you can install individual files. Any 1.2 or later version of Sedona should be sufficient to enable the* ***`Import Sedona environment files`*** *option.*

Installing a device simulator PAR

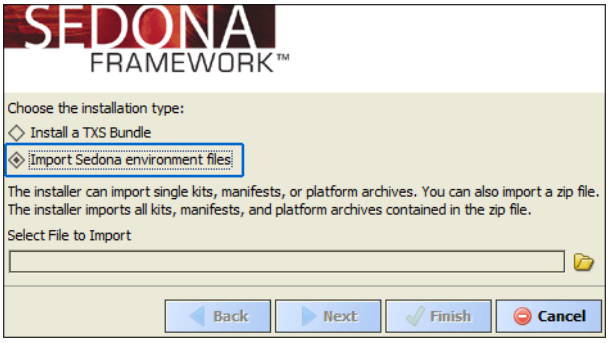
This procedure describes how to install a device simulator platform archive on your PC using the Sedona Installer Workbench tool.

Step 1 In Workbench, select **`Tools > Sedona Installer`**.

Note: *In Sedona TXS-1.2, the Sedona Installer provides the option to* **`Import Sedona environment files`***. This option allows you to import individual kit, manifest, or platform archive files, as well as zip files containing such files.*

Step 2 In the Sedona Installer dialog box, select the option to **`Import Sedona environment files`**, as shown in Figure 1-2, and then click the Directory icon at the right side of the view to open a standard File Chooser dialog box.

Figure 1-2 *Sedona Installer with selected import option*



Step 3 In the File Chooser, locate and select the .PAR file you wish to import. Click **`Open`** to close the File Chooser dialog box.

Step 4 Click **`Next`** and click **`Import`**. On completion, "Import Successful!" displays in the dialog box.

Step 5 Click **`Finish`** to close the **`Sedona Installer`**.

For more details on using the Sedona Installer, see the *NiagaraAX Sedona Framework Installer Guide.*

Now that you have installed a device simulator platform you can begin working with the simulated device.

Chapter 2

Introduction to offline engineering tools

The following topics are covered in this section:

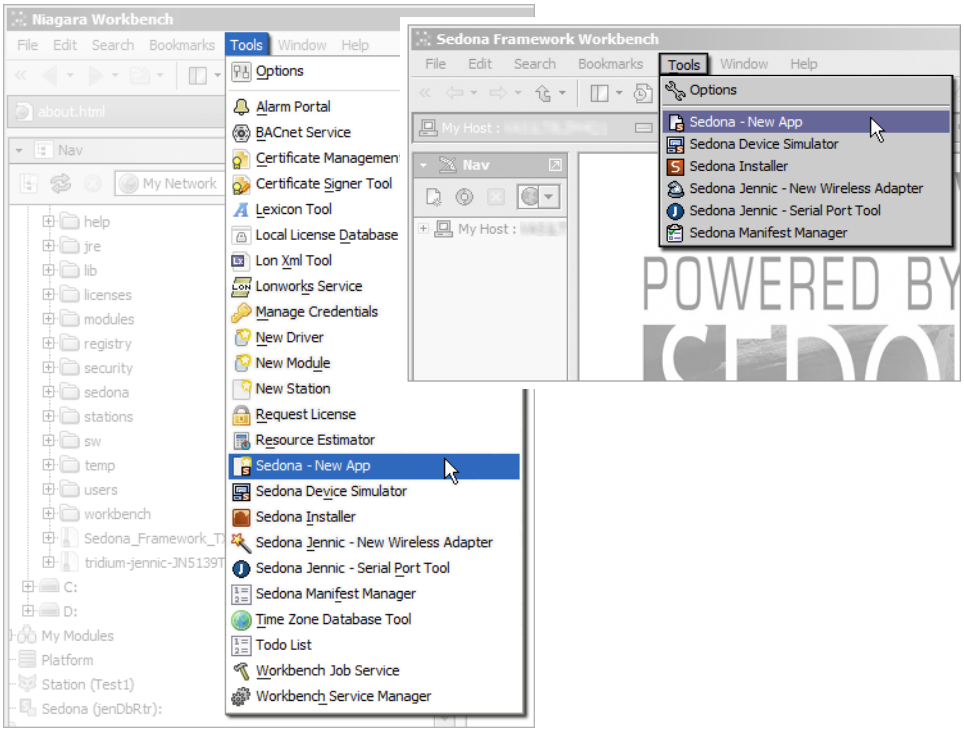
About the Sedona - New App tool

Sedona TXS-1.2 allows you to create Sedona apps in offline mode, similar to the way you create Niagara stations. The tool uses a wizard format much like the New Station tool in Niagara Workbench.

Note: *If using AX-3.7u1 (3.7.106 or later), after installing the TXS-1.2 bundle: version 1.2.2x (for example 1.2.28.4) you need to download and install a patched* `nsedona` *module, version 1.2.28.1. For more details, see the latest NiagaraAX Sedona Framework TXS-1.2 Installer Guide.*

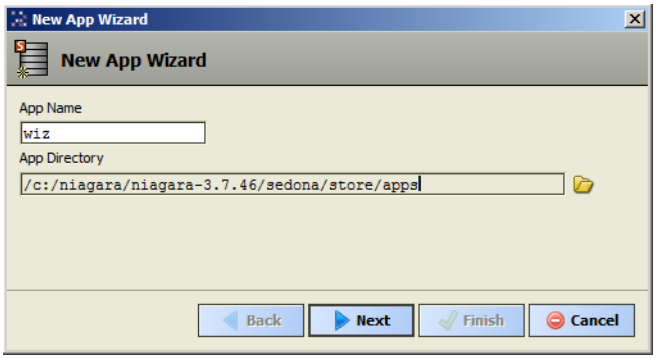
The Sedona-New App tool is available under the Workbench Tools menu when using either NiagaraAX Workbench 3.7 or later enabled for Sedona TXS-1.2, or when using Sedona Framework Workbench 1.2. Both Workbench products provide the identical tool. Access the Sedona-New App tool by selecting the option from the Workbench Tools menu, as shown in Figure 2-1.

Figure 2-1 *Access Sedona - New App tool from Niagara Workbench enabled for Sedona TXS-1.2 (left) and from Sedona Framework Workbench (right)*



Selecting this tool launches the New App wizard which walks you through the steps to create your own Sedona app. The three successive steps allow you to specify the file parameters and access parameters for the app, and to specify additional app parameters to identify the platform the app will run on and the device type.

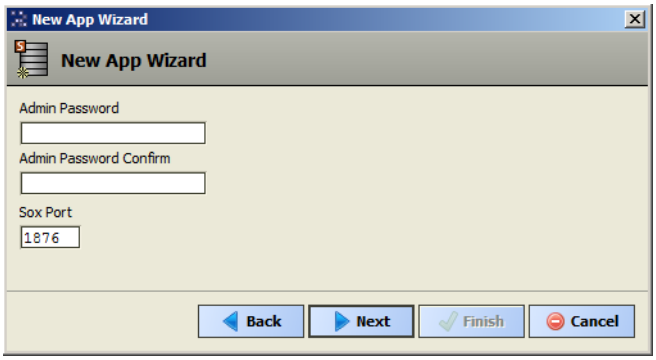
Figure 2-2 *New App Wizard*



First, simply enter the name for the SAX file you wish to create, omitting the file extension. Then click the directory icon to the right of the **`App Directory`** field and use the standard Ord Chooser to select the folder in which you wish to create the app file. Click the **`Next`** button to proceed.

Note: *If the file you specify already exists on your file system, you will be warned and asked if you wish to overwrite the file. Click* **`Yes`** *to overwrite the file or* **`Cancel`** *to go back and enter a different file name.*

Figure 2-3 *New App Wizard*

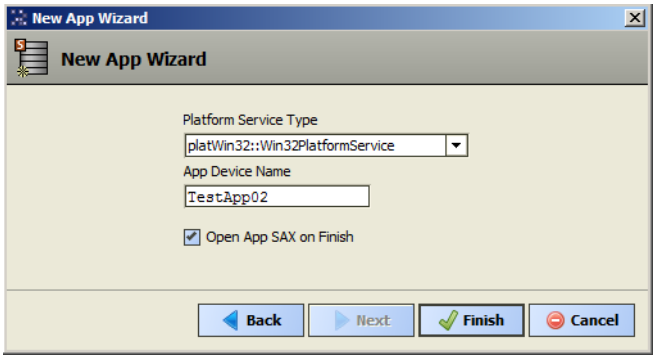


Next, in the **`Admin Password`** field, enter the desired password for the "admin" User. If you do not enter a password, a blank password is used. You must enter the same password in the **`Admin Password Confirm`** field.

Note: *If the two passwords do not match exactly, when you click Next an error message displays to alert you.*

In the **`Sox Port`** field, enter the UDP port for Sox communications. The default, 1876, is pre-loaded for you. Most Sox traffic uses this port.

Figure 2-4 *New App Wizard*



In the last step, specify additional app parameters, and finish the app creation process. In the **Platform Service Type** field, click to display a menu of available platforms and select the one your app uses. All available platform choices are shown in the "Sedona Type" style (kitName::typeName).

The PlatformServiceType drop down is populated with the list of PAR files that you have imported using the Sedona Installer. The platforms may have been imported as part of a bundle, environment file archives, regular PAR files, or simulator PAR files. For details on using the Sedona Installer, see the *NiagaraAX Sedona Installer Guide*.

Note: *If you do not see the platform type that you want in this list, you may need to obtain the platform and install it. To obtain a platform, contact the vendor of that Sedona platform. Otherwise, a sample device simulator platform is included in the Sedona Framework TXS-1.2 Bundle. For more information, see "Sample device simulator platform included in TXS bundle" on page 1-2.*

Finally, you may wish to give the app a "Device Name" in the **App Device Name** field. This will set the root App component's "deviceName" property, which is useful for distinguishing otherwise identical apps from one another in a multiple-device installation.

If you wish to continue editing the app using the Offline App Editor, check the box to **Open App SAX on Finish**, and you will be taken to the offline editor on completion.

Click **Finish** to create the app.

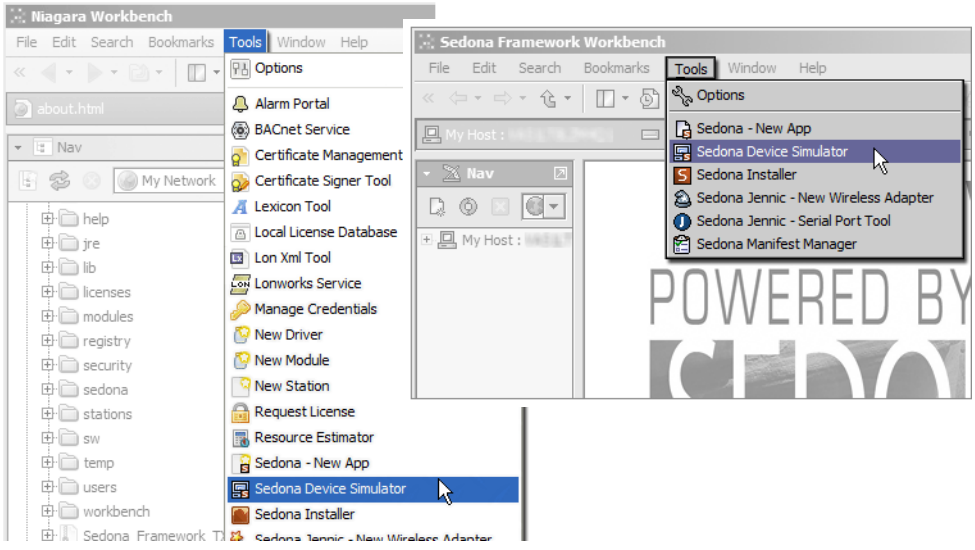
About the Sedona Device Simulator tool

Sedona Framework TXS 1.2 provides the Sedona Device Simulator tool which allows you to run app files on your local PC, including apps that are designed for a different hardware platform. The Device Simulator runs a Sedona Virtual Machine (SVM) that provides a simulated implementation of any native methods that the Sedona kits require.

Additionally, the Sedona Device Simulator allows you to diagnose problems with app logic, and to observe how the app might interact with Niagara, without requiring you to have direct access to the actual hardware on which the app will run.

The Sedona Device Simulator is available under the Workbench Tools menu when using either NiagaraAX Workbench 3.7 or later enabled for Sedona TXS-1.2, or when using Sedona Framework Workbench 1.2. Both Workbench products provide the identical Sedona Device Simulator. Access the Sedona Device Simulator tool by selecting the option from the Workbench Tools menu, as shown in Figure 2-5.

Figure 2-5 *Access Sedona Device Simulator from Niagara Workbench enabled for Sedona TXS-1.2 (left) and from Sedona Framework Workbench (right)*



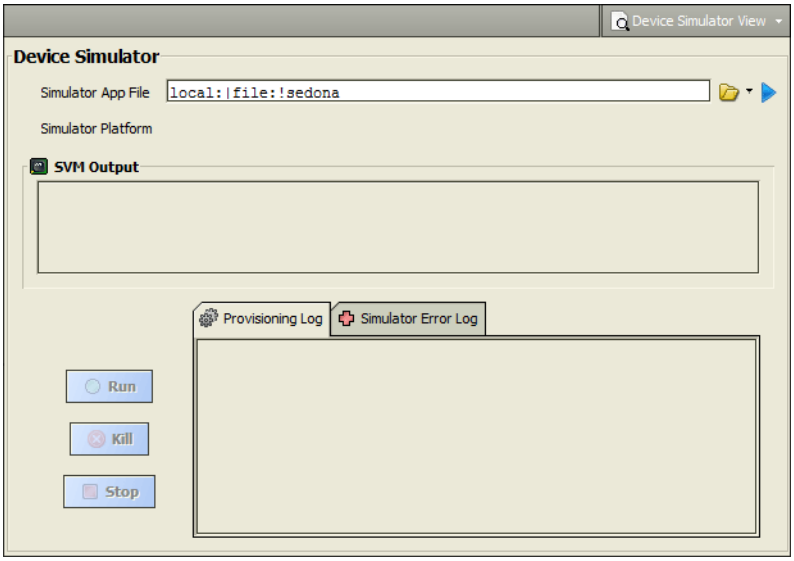
The following topics are covered in this section:

- About the Device Simulator view
- About the Run process
- About simulator app interaction with NiagaraAX

About the Device Simulator view

The Device Simulator view, shown in Figure 2-6, is the default view of the Sedona Device Simulator tool. Use this view to manage the Device Simulator SVM. In the view, you can select an app file to run in the simulated device, as well as choose the simulation platform you wish to use. Once a simulation is running, the view provides device simulator SVM output that is useful for debugging purposes. The view also logs data on the provisioning process and any provisioning errors that occur.

Figure 2-6 *Sedona Device Simulator view*



The Device Simulator view consists of the four areas described below:

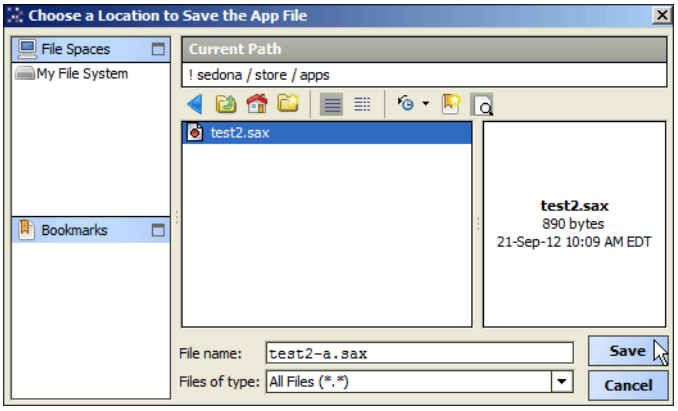
- **Files area**
  Located at the top portion of the view, in this area you will specify the parameters for running the Device Simulator:
  - Simulator App File - Use this field to select the app file you wish to run in simulation mode. The starting location in this field is: `!/sedona`. The file can be in either SAB or SAX format, the tool will convert as necessary. To the right of the field are associated buttons for accessing the File Chooser and viewing the Text File Editor.
    Note: *You cannot choose a file that is part of the platform archive, as this would corrupt the simulator bootstrap file. The tool prevents you from doing this, displaying the following warning: "You cannot provision the Device Simulator with a bootstrap app. Choose an app that is not part of the device simulator database.*
  - Simulator Platform - Use this field to select the device simulator platform that the simulator app runs on. After selecting a Simulator App File, the Sedona platform database is examined and any device simulator archives found that are valid for the selected app file are displayed here. If only one simulator platform is detected, the field is populated with that platform. If multiple simulator platforms are detected this field displays a drop-down option list. In that case, select the simulator platform that represents the device type you intend to simulate.
- **SVM Output area**
  Located just below the files area, the SVM Output area displays the Device Simulator output, just as though you had connected to the debug port of an actual Sedona device running the app.
- **Logs area**
  Located in the bottom half of the view, the Logs area contains two tabs that provide additional information logged about the Device Simulator:
  - Provisioning Log tab - In order to run the selected Simulator App File, the simulator first bootstraps (loads) a default app for that platform, and then uses the Sedona provisioning tools to push the selected app to the platform. The progress of this provisioning process is displayed in this log.
  - Simulator Error Log tab - This log displays any errors that occur with the Device Simulator process during the bootstrapping or provisioning processes.
- **Buttons**
  Located to the left of the logs area, are the control buttons:

- **Run** - Loads the Simulator App File onto the platform.
  This process first bootstraps the Device Simulator with the default app and scode available in the platform archive, and then re-provisions the platform with your selected Simulator App File.
  Note: *The app functions will not affect your Niagara installation. The Simulator is run in a "sandbox" folder called "*`sedonaDeviceSim`*" which is created under your user folder:* `<niagara-release>/users/<username>/sedonaDeviceSim`*). The SVM, scode, and app files are copied to this folder and run from this location. Also, the provisioning process writes files to this location.*
- **Kill** - Stops running the SVM.
  Kill invokes the Sedona App's "Quit" command which first saves the app in the *sandbox* location. Stopping the SVM in this manner will not affect the Simulator App File in the original location.

Caution *If you attempt to resume the simulation by clicking* **Run** *at this point, the app file in the sandbox will be overwritten. The app in the sandbox is overwritten at the start of every* **Run** *process.*

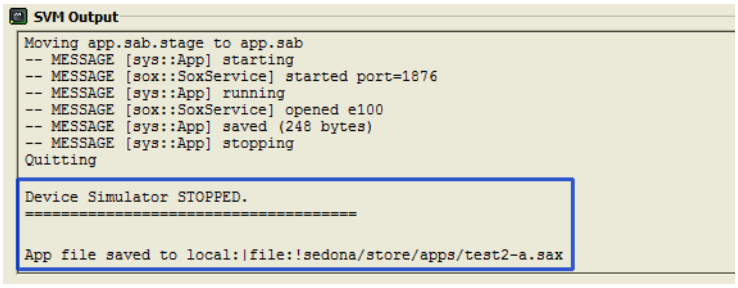
- **Stop** - Stops running the SVM and saves the Simulator App file running in the sandbox location to a file.
  When you click **Stop**, a File Chooser dialog box displays prompting you to choose a location to save the app file, and you can modify the filename, as shown in Figure 2-7. The default filename is the original Simulator App filename that you began with.
  Note: *If the filename you are saving is the same as an existing file, a prompt displays asking if you wish to overwrite the file. If you click* **No**, *the File Chooser displays again. If you click* **Yes**, *the file is overwritten with the revised app saved from the simulator's sandbox.*

Figure 2-7 *File Chooser displays when Stop button is clicked*



Additionally, the SVM Output area shows that the Device Simulator has stopped, shows the filename and the location where the app file is saved, as shown in Figure 2-8.

Figure 2-8 *Resulting SVM Output when Device Simulator is stopped*



About the Run process

Before attempting to run the selected app, the Device Simulator determines if there is an SVM already running that is using the same Sox port, 1876. If so, you will be warned, and asked if you wish to stop the existing app.

Note: *This does NOT check to see if any other apps, stations, etc., are using other ports that the app may require. Also, the simulator app must compete for port control with console-based SVMs that you may have running. So, you cannot run both a console-based SVM and a Simulator SVM on the same port.*

Also, not all behaviors of the physical device may be implemented by the simulation SVM. Behaviors that are typically stubbed out are those that involve interfacing with hardware through inputs, outputs, communication interfaces (i.e. serial ports), indicator lights, and displays.

Note: *When you run an app in the Device Simulator, the app functions occur in the "sandbox" area. This effectively isolates the app functions preventing them from affecting your Niagara installation.*

About simulator app interaction with NiagaraAX

When using the Device Simulator, the simulator always equates to your local host (Workbench), and typically, the same is true for any Niagara station interaction with the app running in the simulator. To obtain Niagara interaction with the simulated device, you can run a local station or JACE station (provided the JACE is on the same network and the Sedona port is open). Once the simulated Sedona device has been added to a Niagara station, proxy points and Px views can be used to check out presentation and operation of the Sedona app. In this manner, you can test the app functionality before actually "replicating" that Sedona device in the station.

About the offline point discovery feature

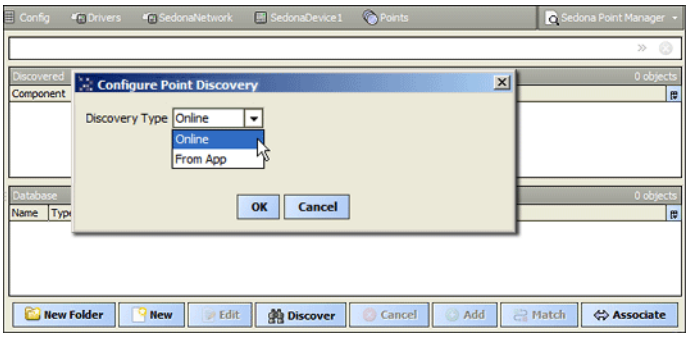
Beginning in NiagaraAX 3.7 enabled for Sedona Framework TXS-1.2, you can perform point discovery on SedonaDevices in an "offline" mode. This allows you to engineer your SedonaNetwork without having to be directly connected to the network of Sedona devices. You can create your station database while sitting in your office, provided you have the app files that are installed in the devices.

This is a simple feature to use. You can use it while editing the station in offline mode, or while connected to a running station. Navigate to the **`Point Manager`** view of the SedonaDevice for which you wish to discover points. Click the **`Discover`** button to initiate point discovery.

Note: *Performing offline point discovery requires that the device be associated with a Sedona app. If the device is not associated with a Sedona app, an error message will appear in the Configure Point Discovery dialog box. If this occurs, cancel the operation and edit the device to associate it with an app.*

At this point, rather than initiating a Sox connection to the device to begin the point discovery, Niagara displays the Configure Point Discovery dialog box. You can choose to perform either "Online" point discovery or "From App" (offline) point discovery, as shown in Figure 2-9.

Figure 2-9 *Point discovery configuration dialog box*



If the station has connectivity, to either a physical device or a simulated device, you can perform the usual direct point discovery by selecting the **`Online`** option.

If you wish to perform offline point discovery against a particular app file, select the **`From App`** option. Click the directory icon to the right of the **`App File Ord`** field to select the app file using the ord chooser, or simply type the app file ord manually.

Note: *When indicating the app file ord for offline point discovery, the file type must be either an XML app file (.SAX) or a binary app file (.SAB). You must use Niagara ord syntax, not Windows file system syntax, for the filename, for example,* `local:|file:!sedona/store/apps/<appFileName>`*. In this example, <appFileName> represents your actual app file name.*

Once you click **`OK`**, the discovery process is identical whether using the **`Online`** or **`From App`** option. You will be presented with the components in the app to allow you to pick points to proxy. If you are not connected to the device at the time, the points will not be able to retrieve values.

Chapter 3

Offline app development

The tasks described in this section demonstrate the typical workflow for offline app development on your local Workbench installation, using either NiagaraAX Workbench 3.7 (enabled for Sedona TXS-1.2) or Sedona Framework Workbench 1.2.

The examples shown here use a sample app file and platform from the device simulator zip file archive included in the TXS-1.2 bundle. For details, see "Sample device simulator platform included in TXS bundle" on page 1-2.

The following procedures are described here:

- Creating a new app in offline mode
- Configuring components in an offline app
- Running an app on a simulated device
- Connecting to the app running on a simulated device
- Modifying the app running on a simulated device
- Making provisioning changes to the app running on a simulated device
- Saving the modified simulator app file

Note: *If you have an app file on hand and have already imported a simulator platform into your Workbench installation, then you can skip to the procedure for Running an app on a simulated device.*

If you have imported a device simulator platform but do not yet have an app file, you can use the **`Sedona - New App`** tool to create an app to run in the simulator. Begin with the following procedure for Creating a new app in offline mode.

Creating a new app in offline mode

This procedure describes how to use the **`Sedona - New App`** tool to create new app in offline mode.

Note: *When creating a new app using the Sedona - New App tool, you must select the correct Platform Service Type in the last step of the app creation process. If using the sample device simulator platform, the correct platform service type is* `jennic::JennicRouter`.

Step 1 In Workbench, click **`Tools > Sedona - New App`** to launch the **`New App`** dialog box.

Step 2 In the **`New App`** dialog box, enter the **`App Name`** and click **`Next`**.

Step 3 Enter credentials for the app, leave the default **`Sox port`** unchanged and click **`Next`**.

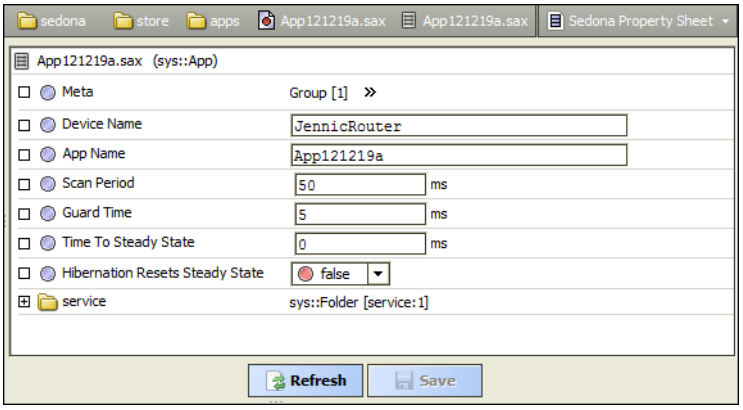
Step 4 Select the **`Platform Service Type`** and enter the **`App Device Name`** and click **`Finish`**, as shown below.

Note: *The PlatformServiceType drop down is populated with a list of available platforms. This list is built from the PAR files which you have imported using the Sedona Installer. The platforms may have been imported as part of a bundle, environment file archives, regular PAR files, or simulator PAR files.*

Figure 3-1 *Creating a new Sedona App*

On completion, the Sedona property sheet for your new app displays.

Figure 3-2 *Property sheet for new offline app*



Once the app is created, you can further develop the new offline app by configuring components for it, such as adding new points offline or making provisioning changes offline.

Configuring components in an offline app

This procedure provides an example of configuring components in a new offline app. Specifically, making provisioning changes and adding new points while working offline.

Prerequisites:

- A new offline app file

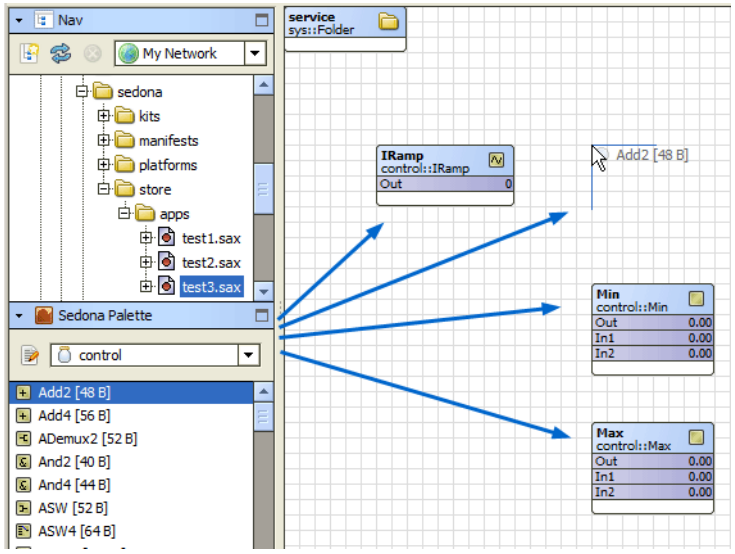
Step 1 Open a Wire Sheet or Property Sheet view for the app.

Step 2 Select **`Window > Side Bars > Sedona palette`**

Step 3 In the Sedona palette, click the `Edit App Schema` icon to add kits to the palette, such as the **`control`**, **`logic`**, and **`math`** kits.

Step 4 Drag and drop components from the palette to add them to the wire sheet, as shown below.

Figure 3-3 *Add components to wire sheet view*



Step 5 Save the modified app file.

Running an app on a simulated device

This procedure describes how to run an app in the Sedona Device Simulator.

Prerequisites:

- An app file
- A simulator platform PAR file (containing a simulator SVM with a default scode image and default "bootstrap" app file) already imported to your Workbench installation.

Step 1 In Workbench, click **`Tools > Sedona Device Simulator`** to launch the tool.

Caution *Do NOT attempt to run the vendor's default "bootstrap" app file in the Sedona Device Simulator. This file is constructed by the platform vendor to provide a basic Sedona app for that platform that, once loaded, is then re-provisioned with your selected Simulator App. If you need a clean App file to start with, create one using the Sedona - New App wizard (for details, see "Creating a new app in offline mode" on page 3-1).*

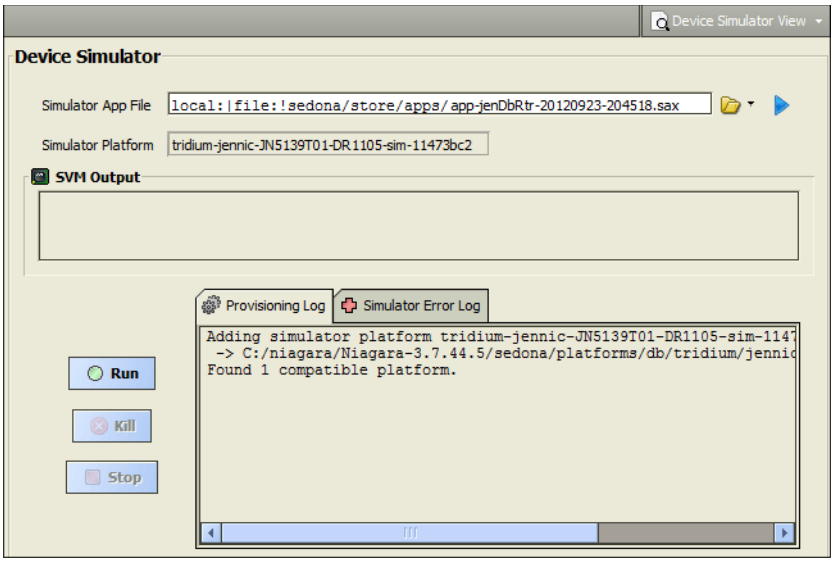
Provisioning the vendor's default "bootstrap" app with a different app file (containing different properties and possibly different kits), either in the Device Simulator, or through a Sedona Tools connection, and then saving this app using the Stop button, will corrupt the vendor's default app file and prevent proper operation of the Device Simulator.

Step 2 In the Device Simulator View, select your app file. Locate your app file by clicking the directory icon (to the right of the Simulator App File field) which displays a standard File Chooser dialog box.

The Device Simulator determines a list of compatible simulation platforms for the selected app. If only one compatible simulation platform is present, it displays in the Simulator Platform field, as shown below.

Note: *If multiple compatible simulation platforms are present, then the simulator populates the Simulator Platform field with a drop-down list of options. You can then select a platform to use in the simulation.*

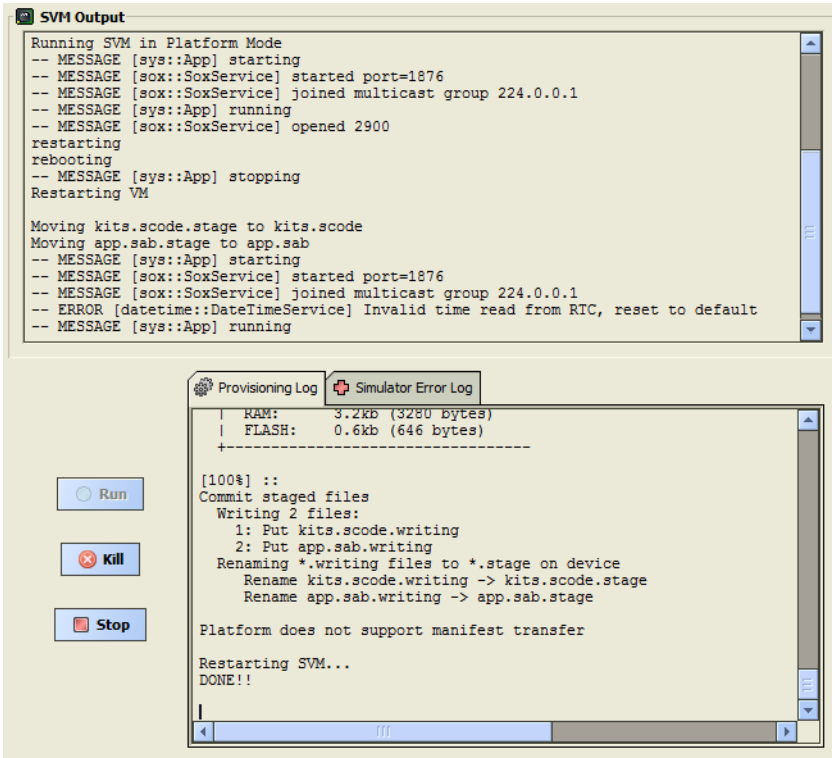
Figure 3-4 *Sedona Device Simulator view with selected Simulator App File and Simulator Platform*



Step 3 In the Device Simulator View, with the Simulator App File and Simulator Platform selected, click **`Run`** to start the simulation.

The **`SVM Output`** pane displays information on the app initialization and re-provisioning process. An example of the typical SVM output on completion of the initializing process is shown here.

Figure 3-5 *Device Simulation Success*



Note: *The Device Simulator runs in a "sandbox" folder called "*`sedonaDeviceSim`*" which is created under your user folder: <*`niagara-release`*>/users/<*`username`*>/*`sedonaDeviceSim`*). The SVM, scode, and app files are copied to this folder and run from this location. The provisioning process writes files to this location.*

Now that the app is successfully initialized and running in the simulator, you can open a Sox connection to the app in the simulated device exactly as you would connect to an ordinary Sedona device running this app, see Connecting to the app running on a simulated device. You might want to make the Sox connection to the simulated device in a separate tab, that way you can easily access the SVM output pane and controls in the Device Simulator.

Connecting to the app running on a simulated device

The procedure describes how to open a direct Sox connection to the app running on a simulated device, in the same manner as you would connect to an ordinary Sedona device running the app.

Prerequisite:
An app file must be successfully running in the Device Simulator.

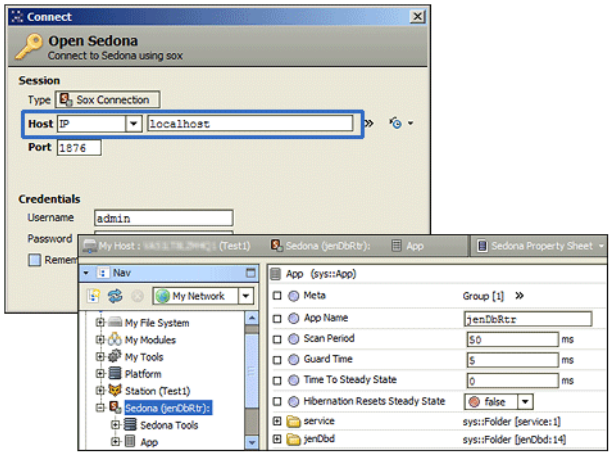
Step 1 In Workbench, open a direct Sox connection using localhost. Select **`File > Open > Open Device`**.

Step 2 In the Connect dialog box, for **`Host IP`**, enter: `localhost`.
In the **`User name`** and **`Password`** fields, enter the credentials used in the new app file that you created, and click **`Finish`**.

Note: *If using the sample app from the sample device simulator download, enter these credentials:*
`Username:` `admin`
`Password:` `<blank>`.

On completion, you have a direct Sox session connection to the app running in the simulated device, as shown here.

Figure 3-6 *Open Sox connection to app running in Device Simulator*



Modifying the app running on a simulated device

This procedure provides an example of viewing and modifying the app running on a simulated device. The example shows typical app logic modifications that you can make, such as adding points.

The sample app used here is developed to run on a device that has four leds but currently controls only two of those leds. The example shows steps to add two more digital outputs to control the remaining two leds.

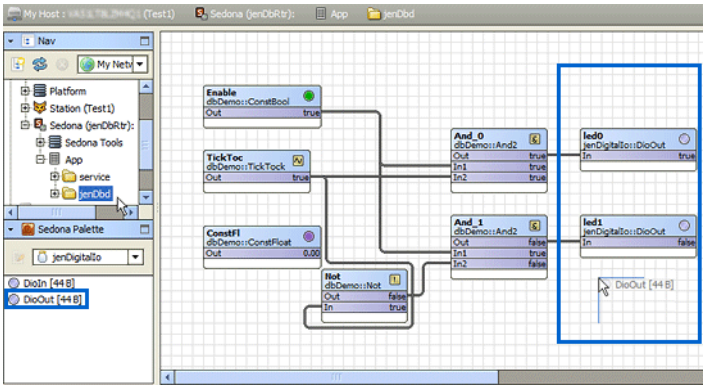
Note: *Interim changes to the simulator app are actually applied to a copy of the app file that is running in the Device Simulator's sandbox area. When you end the simulation using the Stop button, you are prompted about where to save the modified app file. For more information about the sandbox area, see "About the Run process" in the section on "Running an app on a simulated device" on page 3-3.*

Prerequisites:

- The app must be successfully running in the Device Simulator.
- A Sox connection exists to the app currently running on the simulated device.

Step 1 In the Nav tree under **`MyHost`**, expand the **`Sox device`** node and the **`App`** node. Double-click the **`JenDbd`** component to display the wire sheet view, as shown below.

Figure 3-7 *Modifications to app logic in wire sheet view of app component*



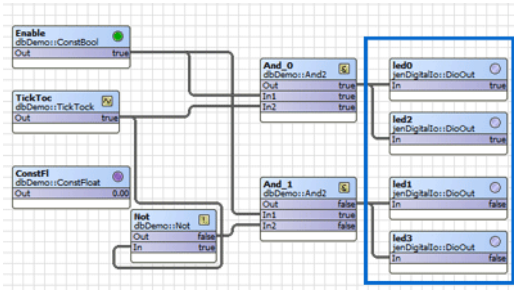
Step 2 In the Sedona palette, open the **`jenDigitalIO`** palette and drag two **`DioOut`** objects to the wire sheet, naming them "`led2`" and "`led3`".

Step 3 In the wire sheet view, link the `And_0` output to the `led2` input.

Step 4 In the wire sheet view, link the `And_1` output to the `led3` input.

After making this change, the app lights leds 0 and 2 when `And_0` is true, and lights leds 1 and 3 when `And_1` is true, as shown.

Figure 3-8 *Added digital IO controls to the app*



Making provisioning changes to the app running on a simulated device

This procedure provides an example of making provisioning changes in the app running on a simulated device using the Kit Manager provisioning tool under the device to add a kit to the app.

Prerequisites:

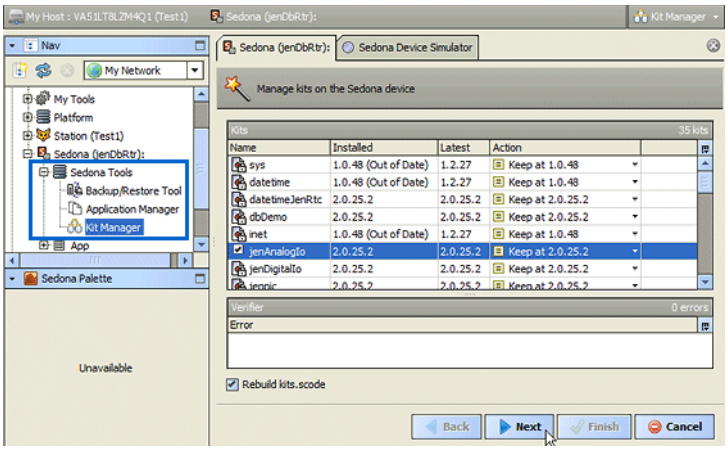
- The app must be successfully running in the Device Simulator.
- A Sox connection exists to the app currently running on the simulated device.

To add the **`jenAnalogIo`** kit to the app:

Step 1 In the Nav tree, expand the **`Sedona Tools`** node under the Sox device and double-click **`Kit Manager`** to launch the provisioning tool.

Step 2 In the Kit Manager view, click the checkbox to select the **`jenAnalogIo`** kit, as shown below, and click **`Next`** and **`Finish`**.

Figure 3-9 *Use Kit Manager to add jenAnalogIo kit to the app*

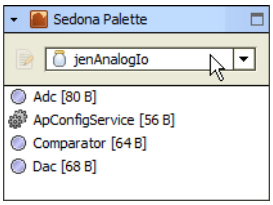


Step 3 When kit changes have successfully completed, click **`Restart`** to start the device.

Step 4 In the Nav tree, double-click the Sox device node to reconnect to the app running in the Device Simulator.

Step 5 Access the wire sheet view for the **`JenDbd`** component in the app, and in the Sedona palette, open the **`jenAnalogIo`** palette, as shown here.

Figure 3-10 *Added kit provides the jenAnalogIo palette*



This change makes the palette of Jennic analog input/output controls available for use in app logic modifications.

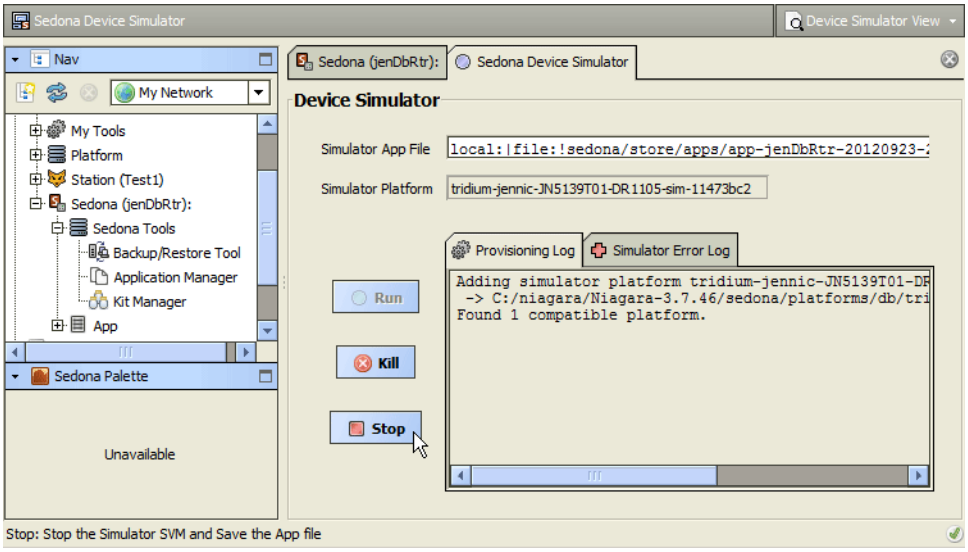
When finished making changes to the app running in the Device Simulator, you can stop the simulation and save the modified app to your file system.

Saving the modified simulator app file

This procedure describes how to stop the simulation and save the modified app to your file system.

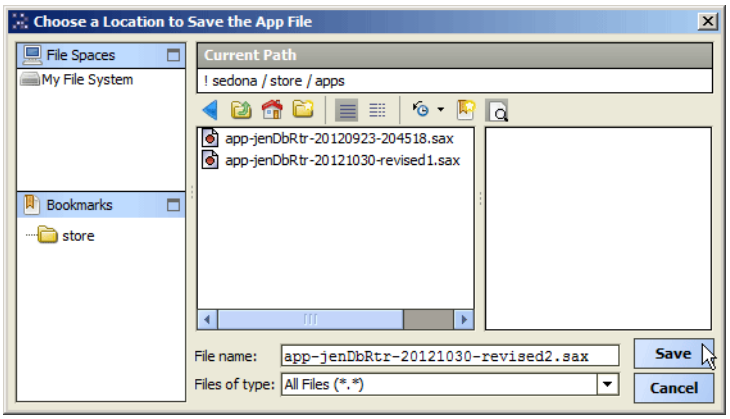
Step 1 In the Sedona Device Simulator, while the app is running, click **`Stop`**, as shown below.

Figure 3-11 *Click Stop to stop the simulator SVM and save the modified app*



A File Chooser dialog box displays prompting you to choose a location to save the app file, as shown below. The default location is the `!sedona/store/apps` folder. The default file name is the original Simulator App file name that you began with. Change the file name as needed to avoid overwriting the original app file.

Figure 3-12 *File Chooser displays when Stop button is clicked*



Note: *If the filename you are saving is the same as an existing file, a prompt displays asking if you wish to overwrite the file. If you click* **No**, *the File Chooser displays again so you can modify the filename. If you click* **Yes**, *the file is overwritten with the revised app saved from the simulator's sandbox.*

On completion of this step you successfully stopped the simulation and saved the modified app to your file system.

Note: *If you encountered any problems while working with the simulated device, see the "Troubleshooting" appendix.*

Chapter 4

Offline network engineering

Using NiagaraAX Workbench 3.7 (enabled for Sedona TXS-1.2), you can engineer Sedona networks in offline mode. All offline engineering tasks described below can be completed on the PC in your office, without connecting to the actual device network on-site. You can manually add new devices and begin station engineering for a Sedona network. You can also connect to a simulated device and use matching to "sync" to existing components (device-level, proxy points, and so forth) in the device database.

The following topics are covered in this section:

About app file associations

An important aspect of offline engineering involves app file "associations" with Sedona device components which, among other things, enables offline point discovery. With Sedona TXS-1.2, all Sedona environment files (kits, manifests, and platform archives) and Sedona app files for Sedona devices networked under a JACE are stored and maintained on that JACE platform. Having all these files on the JACE allows many functions to be initiated from the JACE station, such as Sedona provisioning and Sedona point discovery which is done from the stored Sedona app file. Also, having associated Apps on the JACE allows them to be included as part of the station backup. The act of getting a device's app file onto the JACE and connected with a Sedona device component that represents a specific physical Sedona device is called *association*. For more details, see "Sedona device Association" in the *Sedona Framework TXS-1.2 Networks Guide*.

Note: *When duplicating devices that have app file associations, you must correct the* `deviceName` *and* `appName` *properties within the duplicated devices. For details, see "About duplicating devices that have app file associations" on page 5-1.*

Typical workflow for offline engineering

This section demonstrates the typical workflow for engineering Sedona networks in offline mode.

The examples shown here use the app file and platform included in the sample device simulator download file available on the *Niagara-Central* portal. For details, see "Sample device simulator platform included in TXS bundle" on page 1-2.

Caution *There are known limitations to using the sample device simulator platform to engineer a SedonaJen6LpNetwork offline. For details, see "Limitations engineering Jen6Lp networks using the simulator" on page 1-2*

The following procedures are described in this section:

Discovering the simulated device and adding it to a locally running station

This procedure describes performing device discovery against the device simulator and adding the discovered device to a Sedona Network in a locally running station. Configuring this new device establishes the device's app file association and places a copy of the app file in the station database.

Note: *For more details, see "Configure SedonaNetwork" in the* NiagaraAX Sedona Framework TXS-1.2 Networks Guide*.*

Prerequisites:

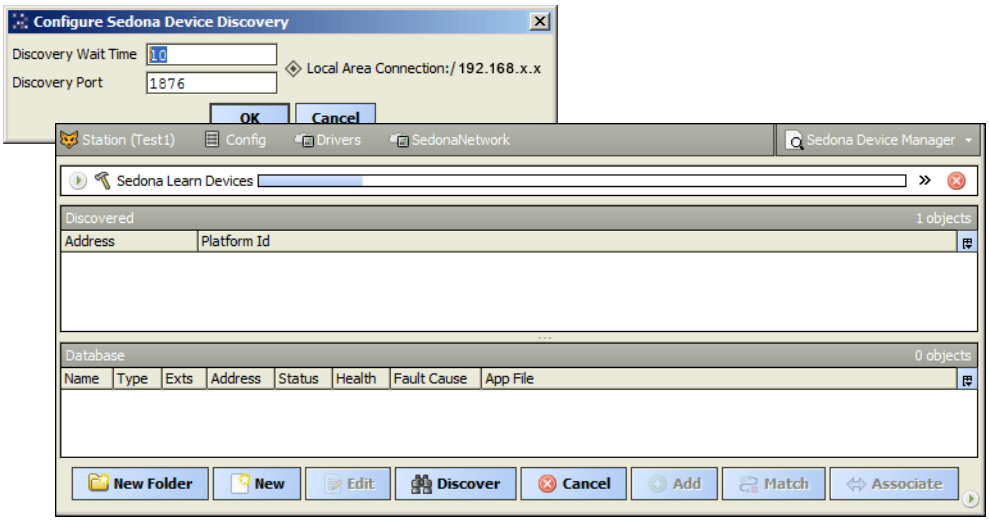
- An app file successfully running in the Device Simulator
- A locally running station with a Sedona Network

Note: *You must to be licensed for* `sedonanet` *in order to run a Sedona Network.*

Step 1 In the nav tree under `MyHost`, locate and expand the a locally running station containing a Sedona Network. Expand the **`Config`** node. Double-click on the **`Drivers`** node to display the **`Driver Manager`** view.

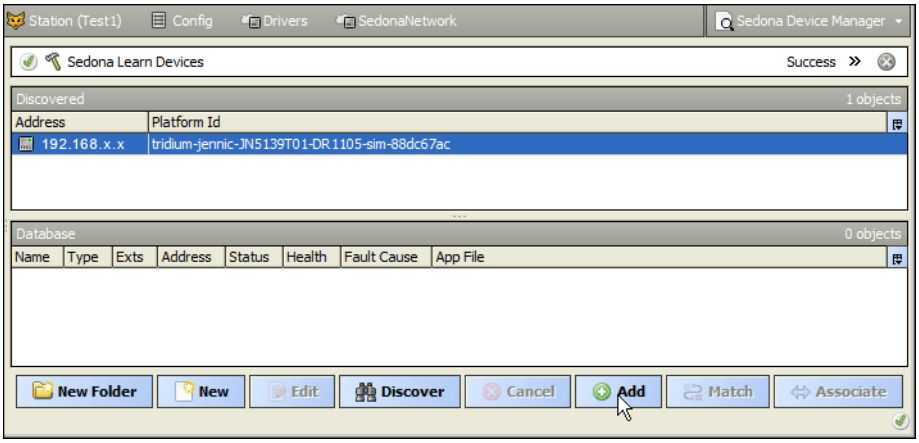
Step 2 In the **`Driver Manager`** view, double-click the Sedona Network to open the **`Sedona Device Manager`** view, and click **`Discover`** at bottom of the view to initiate device discovery against the device currently running in the simulator. The Configure Device Discovery popup displays showing the IP address for the local host, click **`OK`**. The Learn Devices progress bar displays while the simulated device is detected, as shown here.

Figure 4-1 *Discover the simulated device*



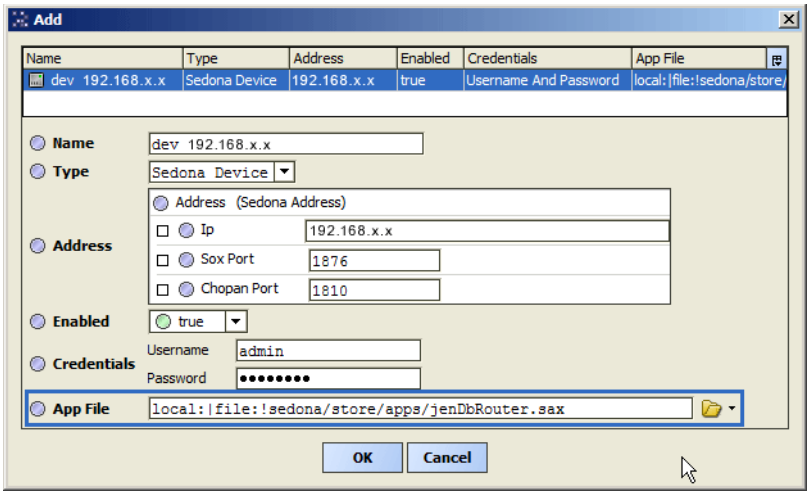
Step 3 In the **`Sedona Device Manager`** view, select the discovered device displayed in the **`Discovered`** pane and click **`Add`**.

Figure 4-2 *Add discovered device to station*



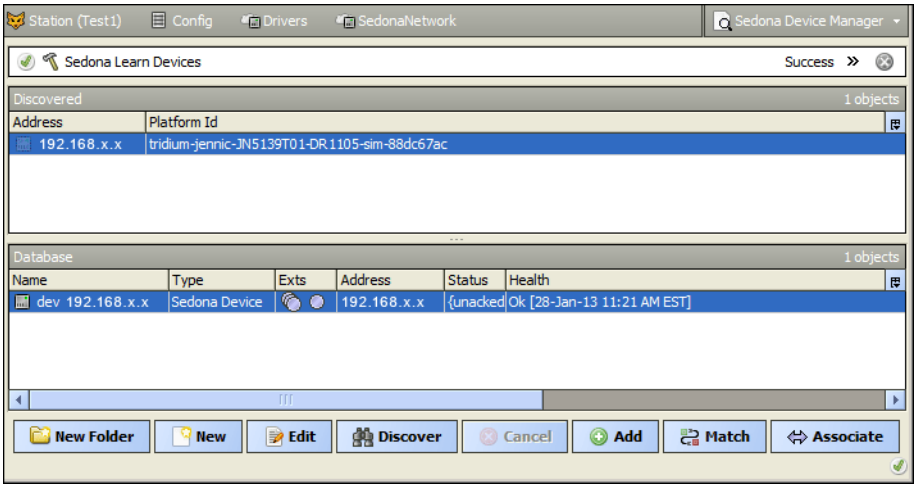
Step 4 In the **Add** dialog box, click the Directory icon to the right of the **App File** field to select the app file currently running in the device simulator and click **OK**, as shown below. This establishes the app association for the device, placing a copy of the simulator app file in the station directory database.

Figure 4-3 *Add dialog box shows discovered device, App File field shows selected simulator app file*



The added device displays in the **Database** pane in the lower half of the Sedona Device e Manager view, and the device also appears under the SedonaNetwork node in the Nav tree.

Figure 4-4 *Discovered device added to station database*



Performing point discovery on the simulated device

This procedure describes how to discover data items in the simulated device as point candidates for inclusion in the station database which can be used as binding targets for widgets in a Px View, alarm and history extensions, and links to schedules and control logic.

Note: *For more detailed information on discovering proxy points, see "Create Sedona proxy points" in the NiagaraAX Sedona Framework TXS-1.2 Networks Guide.*

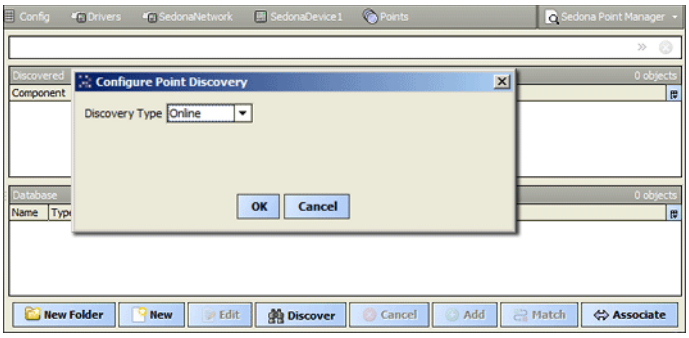
Prerequisites:

- An app file successfully running in the Device Simulator
- A station running locally with a Sedona Network
- A new Sedona device associated with the app file currently running in the Device Simulator

Step 1 In the new Sedona Device, open the **Sedona Point Manager** view and click **Discover** to initiate point discovery against the device currently running in the simulator.

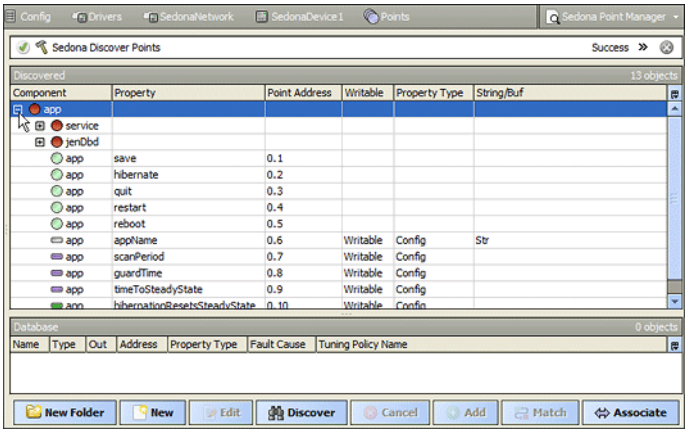
Step 2 In the **Configure Point Discovery** dialog box, the default discovery type is **Online** when online with the simulated device. Click **OK** to initiate the discover.

Figure 4-5 *Configure point discovery*



After completing point discovery, expand the app node to see the point candidates, as shown here.

Figure 4-6 *Discovered Sedona component slots in the sample app*

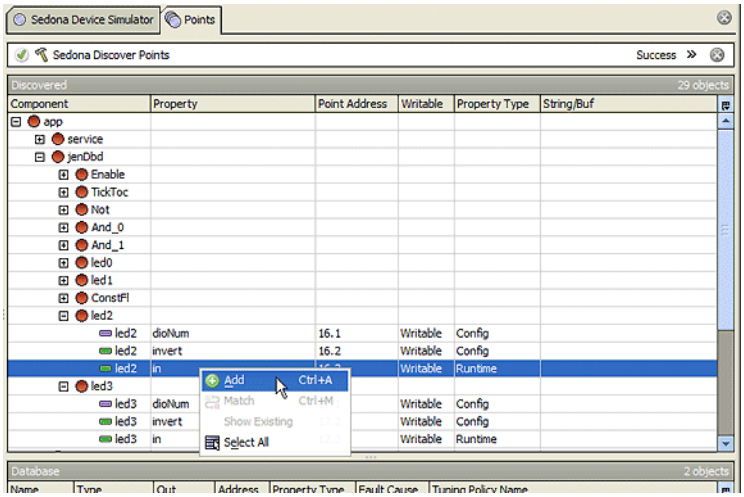


Step 3 In the Discovered pane, right-click a point and click **Add** to add a new proxy point.

Using this method, the proxy point added to the database is matched to the point in the device, configured with the appropriate `Address` value.

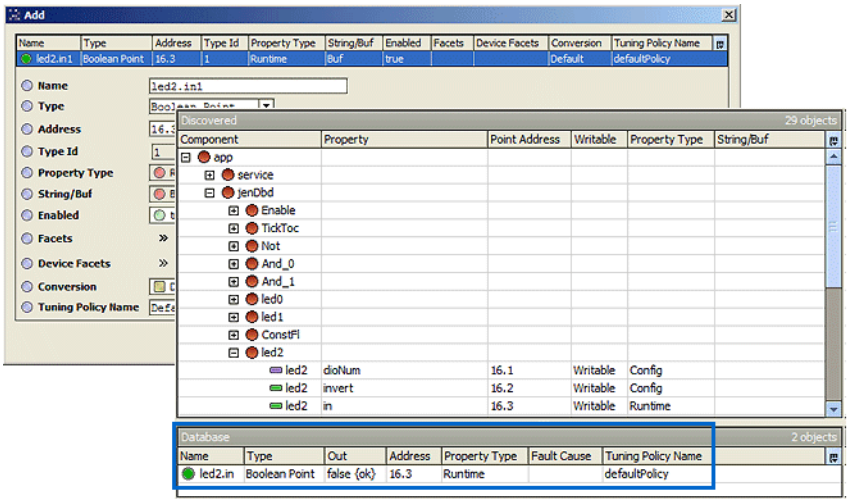
Note: *When working off-site, the recommended method for creating proxy points is to right-click a point in the discovered pane and click* **Add**. *"Adding" configures the proxy point with the correct* `Compid:Slotid` *address value. Other methods for adding new proxy points, such as using the New button or the palette, are not recommended.*

Figure 4-7 *Added new proxy points in the database*



Step 4 When you have the proxy point(s) configured for your usage, click **OK**. The proxy points are added to the station, and appear listed in the Database pane, as shown below. Within a few seconds, the points should reflect current values. Of course, these values will be static because you are running against a simulator SVM.

Figure 4-8 *Added new proxy point in the database*



Viewing proxy points to confirm expected behavior

This procedure describes how to view proxy points to see points, such as ramps, updating as polling occurs.

Step 1 In the Nav tree under the expanded SedonaDevice, right-click **Points** and click **Views > Wire Sheet** to display the wire sheet view.

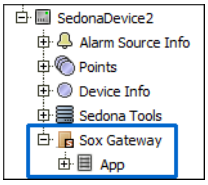
Step 2 Confirm that the proxy point values are updating.

Accessing the simulator app via the Sox Gateway

This procedure describes how to access the app in the simulated device in order to modify app logic.

Step 1 In the Nav tree, expand the Sox Gateway node under the device. Then right-click the **App** node and select **Views > Wire Sheet** to display the wire sheet view for the app, as shown here.

Figure 4-9



Modifying the simulator app, saving changes and updating station file space

This procedure provides an example of making changes to the app in the simulated device which you have accessed via the Sox Gateway, updating the app in the station file system by performing a Get, as well as saving the modified app to your file system.

Note: ***When you modify the app running in the simulated device, those changes are not reflected in the copy of the app located in the station. You must use the Application Manager provisioning tool to Get the updated app from the simulated device and place a copy in the station file space. Doing this ensures that the most recent version of the app file is placed in the station file space.***

To remove points from the app:

Step 1 In the Nav tree, access the Wire Sheet view for app, select two of the point components, `LED0` and `LED1`, right-click and click **Delete**.

Step 2 In the Nav tree, expand the **Sedona Tools** node and double-click the **Application Manager** to launch the provisioning tool.

Step 3 Perform a **Get** operation which accomplishes two things:

- Saves the changes to the simulator app file running in the sandbox area for the Device Simulator
- Places a copy of the modified app file in the station file space, updating the app file there

Step 4 In the Device Simulator, click **Stop** to end the simulation and save the modified app (in the sandbox area) to your file system, in the `!sedona/store/apps` directory.

Note: *If you encountered any problems while working with the simulated device, see the "Troubleshooting" appendix.*

The completed app can now be transferred to the actual job site for installation.

Chapter 5

Engineered station installation

Using NiagaraAX Workbench 3.7, enabled for Sedona TXS-1.2, you can install the engineered station at an actual job site.

The following topics are covered in this section:

About duplicating devices that have app file associations

If you have associated an app with a device, and then duplicated that device within the station, you must correct the associations in the duplicated device. By default, a duplicated device is associated with the app in the original device. Since each device must be associated with its own, unique app file, the association must be corrected. To correct this, make a copy of the original app, change certain properties within the copy, and then associate this revised app with the duplicate device.

Within each .SAX app file, there are properties that define the device name and app name of its associated device. These properties (`deviceName` and `appName`) are configured when you associate a device with the app. When you make a copy of the original app file, these are the properties that must be changed in the copy to associate the duplicate device name and new app name.

For example, let's assume App1 in your station is associated with Device1 and you want to reuse that app in a second device. So, you duplicate Device1, naming the duplicate device, Device2. Then, make a copy of App1, naming it, App2. Finally, correct the associations in the file named App2, changing the `deviceName` property from Device1 to Device2, and the `appName` property from App1 to App2.

Figure 5-1 *App file association properties*

```
<sedonaApp>
<schema>
  <kit name="sys" checksum="d3984c51" />
  <kit name="control" checksum="808b7db3" />
  <kit name="inet" checksum="25648ba7" />
  <kit name="logic" checksum="9fe95ce1" />
  <kit name="math" checksum="c22b255c" />
  <kit name="platWin32" checksum="3746c8c1" />
  <kit name="sox" checksum="397a84dd" />
</schema>
<app>
  <prop name="deviceName" val="device1"/>
  <prop name="appName" val="app1"/>
  <!-- /service -->
```

Do this for all duplicated devices. If you plan to reuse an app in 25 duplicated devices, you must make 25 copies of the original app file, renaming and editing each copy of the file, and correcting the associations of those app files with the appropriate duplicated device.

Typical workflow for installing the engineered station at the job site

This section shows the typical workflow for installing the engineered station at an actual job site:

Prerequisites:

- JACE commissioned with NiagaraAX 3.7 enabled for Sedona Framework TXS 1.2
- At least one Sedona device

Transferring the station to the remote JACE

This procedure gives the basic step for transferring the station (that was engineered off-site) to the JACE.

Step 1 Use the Station Copier platform tool to install the new station on the remote JACE. For details on copying stations, see "Station Copier" in the *NiagaraAX Platform Guide*.

Installing the vendor-specific files on the JACE

This procedure gives the basic step for installing the vendor-specific files in the Sedona environment on the JACE using the Sedona Environment Manager platform tool.

Step 1 Once the new station is running on the JACE, connect to the remote station, and use the Sedona Environment Manager platform tool to install the vendor-specific files (kits, manifests, and platforms) from your local Workbench installation to the JACE.

For complete details on installing the vendor-specific files on the JACE, refer to "Installing Sedona environment files" in the *NiagaraAX Sedona Framework TXS Networks Guide*.

Performing device discovery, match, and associate

This procedure describes performing device discovery, matching the physical device to the device in the station database, and associating the appropriate app which adds the associated app to the JACE station file space.

Note: *Device discovery will only work when the correct type of Sedona network has been selected. For Ethernet LAN devices make sure to select 'SedonaNetwork' when adding a network to your Niagara station. Regarding Jennic-based (wireless) devices, see "Limitations engineering Jen6Lp networks using the simulator" on page 1-2*

Step 1 Power up the physical Sedona device.

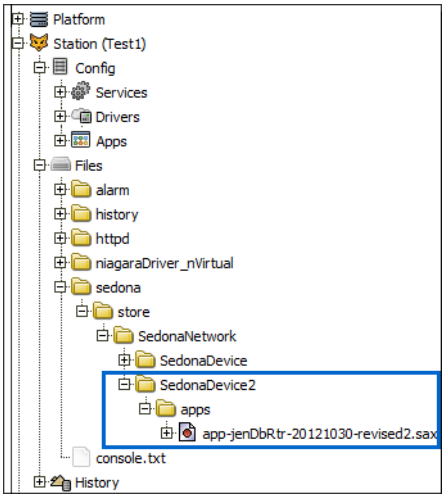
Step 2 In Workbench, open the Sedona Device Manager and click **`Discover`** to detect the physical device and display the device in Discovered pane.

Step 3 Select the discovered device and click **`Match`** to match the device to the device in station database.

Step 4 Select the discovered device in the Discover pane and the added device in the Database pane, then click **`Associate`**. A File Chooser window displays.

Step 5 In the File Chooser, navigate to the offline app file in `!sedona/store/apps` and click **`Open`**. This step adds the indicated app file to the JACE station file space, as shown below, using a file system hierarchy that includes the device's name. This facilitates retrieving the app file later for point discovery, provisioning with app put, etc.

Figure 5-2 *Associating the app file adds associated device to station file space*



Pushing the offline app down to a physical device

This procedure describes using the a Put operation in the Application Manager provisioning tool to copy the associated app file from the JACE file space to the physical device.

Step 1 Expand the Sedona Tools node under the device in the station.

Step 2 Double-click the Application Manager provisioning tool

Note: *The app file copied to each device must be associated with that particular device.*

For complete details on provisioning the device, refer to "Put an application on a device" in the *Sedona Framework TXS Tools Guide*

Step 3 Click **`Put`** and **`Next`**, and select the appropriate associated app to install on the device.

Step 4 Click **`Finish`** to push the selected app down to the physical device.

Viewing points to confirm behavior

The procedure gives the basic step to confirm app logic is working as intended.

Step 1 Within device in the station database on the JACE, examine the Points in the Wire Sheet view to confirm that the proxy point values are updating.

This concludes the process for installing the engineered station at an actual job site.

Appendix A

Troubleshooting

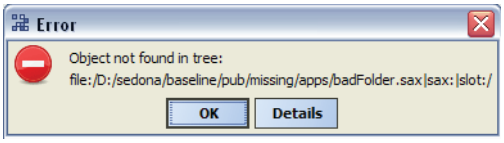
Troubleshooting tips for problems that you may experience while developing in offline mode are covered in this section.

Offline app creation errors

Specified directory structure that does not currently exist

If you have specified a directory structure that does not currently exist on your file system, the wizard will create the necessary intermediate folders to generate the app. However Niagara will not be able to sync the tree at first, and you may see the following error dialog. Simply click **OK** to close the dialog box.

Figure A-1 *New app creation error*



Admin password and admin password confirmation do not match

When creating a new app, after clicking Finish, you may see an error dialog indicating that the password is not confirmed. This means that the entries for admin password and admin password confirmation do not match. This error can occur (even if the entries actually do match) when using AX-3.7U1 and an unpatched `nsedona` module. If this is the case, you need to download and install the patched `nsedona` module, version 1.2.28.1. For more details, see the latest *NiagaraAX Sedona Framework TXS-1.2 Installer Guide.*

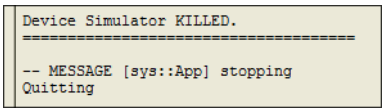
Figure A-2 *Admin password and confirmation do not match error*

Device simulation errors

Determine if the SVM has stopped running

In most cases, when the device fails it will stop the running SVM. An example of this would be if the SVM "bootstraps" the default app but then fails to provision it with your Simulator App. You should confirm that the SVM has stopped running in the SVM Output pane by verifying that you see the message shown in Figure A-3.

Figure A-3 *SVM Output pane shows SVM has stopped running*



Common failure messages

The following common failure messages issued by the device simulator, as well as the likely cause and solution, are covered in this section:

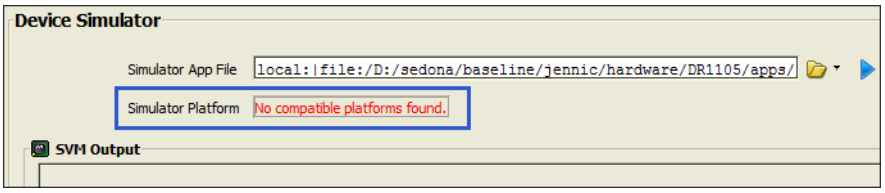
- No compatible platforms
- Missing kit
- Missing manifest
- Not a valid app
- App contains validation error
- Another SVM is running

Problems not listed here may be network issues. For information on Sedona Framework networks, refer to the *Niagara AX Sedona Framework Networks Guide*.

No compatible platforms

If your Simulator Platform field displays the message shown in Figure A-4, then you do not have any Simulator Platform archives that provide a simulator SVM for the selected Simulator App File.

Figure A-4 *No compatible platforms error message*

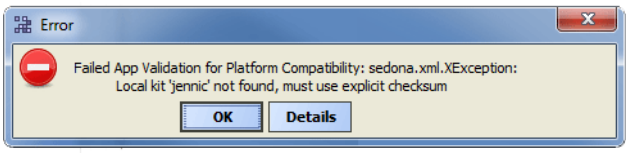


To correct this problem, contact the vendor of the platform that your App specifies, and ask them to provide you with a simulator archive for that platform. Once you install this into your Sedona platform database you should see the proper Simulator Platform choice.

Missing kit

If you see the message shown in Figure A-5, most likely you do not have a necessary kit in your `SEDONA_HOME/kits` folder. This occurs when you do not have any versions of the required kit. The kits in your local database are organized into folders in the `/kits` directory where each folder name is based on a kit name. This error occurs because you do not have a folder for a specified kit.

Figure A-5 *Missing kit error message*

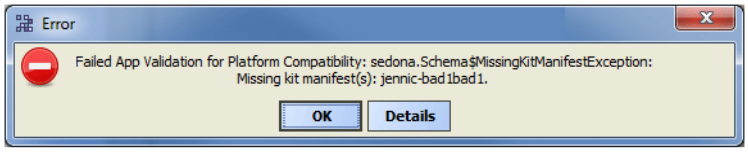


To correct this problem, install the missing kit with the necessary version (if specified in the app file's <schema> section) into a folder bearing the same name as the kit. Place this folder under the `SEDONA_HOME/kits` folder.

Missing manifest

If you see the message shown in Figure A-6, most likely you do not have the necessary *version* of a particular kit in your `SEDONA_HOME/kits` folder. This occurs when you might have one or more versions of a particular kit installed in your Sedona Framework TXS installation, but the app you are trying to run in the Device Simulator specifies a version of the kit that you do not have. You will need that particular version to run this app.

Figure A-6 *Missing manifest error message*



Note: *You may be able to get the app to run by removing the* `checksum="xxxxxxxx"` *text in the* `<schema>` *section of the app. This will allow another version of the same kit to be used.*

Caution *If you remove the checksum text in the schema and there are significant differences in the implementation of the specified version and the version you use, the app may not function correctly to provide the behavior you wish to simulate!*

To correct this problem, install the missing version of the specified kit in to the proper folder in the `SEDONA_HOME/kits` folder.

Not a valid app

If you see the error message shown in Figure A-7, the ord specified in the Simulator App File field is not a valid Sedona app.

Figure A-7 *Not a valid app error message*

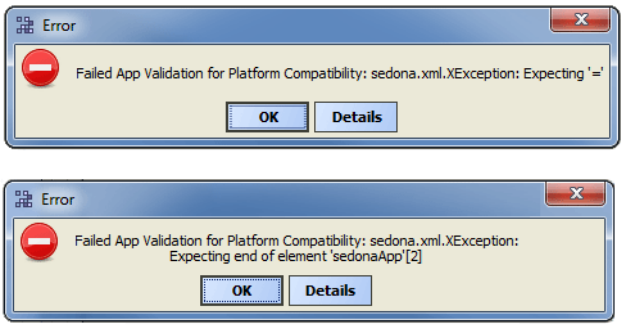
Note: *Specify either a .sax or .sab file. No other file formats are valid.*

To correct this problem, select a valid app file.

App contains validation error

If you see the error messages shown in Figure A-8, the app file contains some sort of validation problem. The specific problem with the app will be included in the error message, as well as the exception details. These examples illustrate two different parsing problems that you might see if the SAX file you are using is not a properly formatted Sedona Application XML file.

Figure A-8 *Two sample app validation error messages*

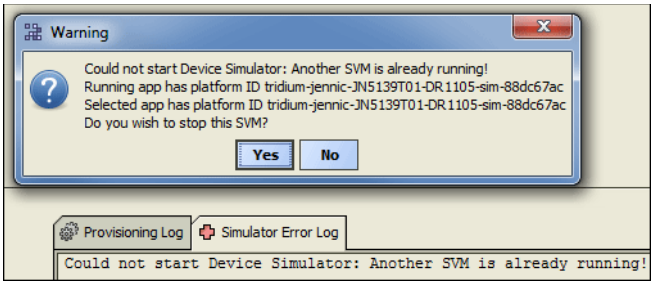


To correct this problem, check your app file to make sure that it is not corrupted, and that it contains valid Sedona XML.

Another SVM is running

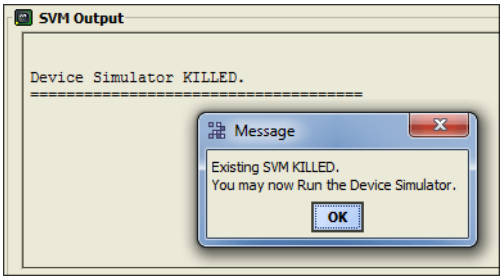
Before starting, the Device Simulator checks for the existence of an SVM that is already running on the desired port. If one is found, you will see the error message shown in Figure A-9. The message informs you that there is already an SVM running, and asks if you wish to stop it.

Figure A-9 *Another SVM is running error message*



To correct this problem, click **`Yes`** or **`No`**, depending on what you wish to do. If you select **`No`**, then the display returns to the Device Simulator view as if you had just selected your app file. If you select **`Yes`**, the tool attempts to invoke the "Quit" command on the app that is currently running. If this is successful, you will see the dialog shown in Figure A-10.

Figure A-10 *Existing SVM killed dialog*



If it is unable to stop the SVM, you can use your operating system Task Manager to determine whether you there is still an "`svm.exe`" process running, and if so, kill the process. Then you should be able to run the Device Simulator.

Offline point discovery errors

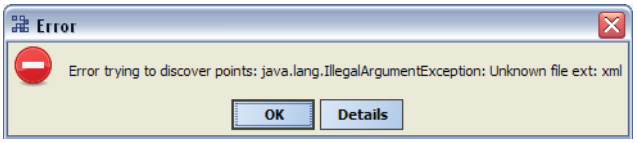
The following point discovery error messages and solutions are covered in this section:

- Specified incorrect file type
- Missing kit manifests
- Missing kit version
- Device not currently associated with a Sedona app

Specified incorrect file type

If you specify an incorrect file type, you will receive the error message shown here. Valid file types for offline point discovery are either an XML app file (.SAX file extension) or a binary app file (.SAB file extension).

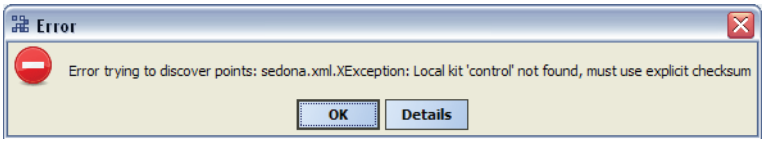
Figure A-11 *Incorrect file type error message*



Missing kit manifests

Niagara AX Workbench uses the kit manifests it has installed to decode the app file and present its components. If you do not have the kits required by the app, you will receive the error message shown here. Use the Workbench Manifest Manager tool to install the missing kit manifests. This will copy manifests from the Workbench machine to the station machine. If you don't have the necessary manifests on the Workbench machine, then you just need to contact the vendor of the kit to get them.

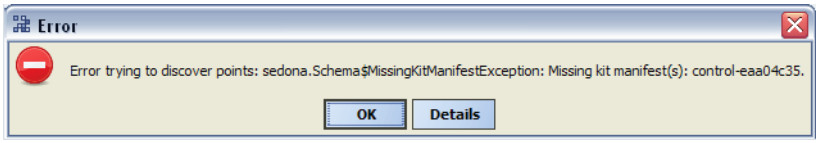
Figure A-12 *Missing kit manifests error message*



Missing kit version

If the app specifies a particular version of a given kit, which you do not have installed in your Workbench you will receive an error message that lists the missing kit version, as shown here. Use the Workbench Manifest Manager tool to install the missing kit versions.

Figure A-13 *Incorrect file type error*



Device not currently associated with a Sedona app

If the device is not associated with a Sedona app, the error message shown here displays. Cancel the operation and edit the device to associate it with a Sedona app file.

Figure A-14 *Incorrect or missing app file association*