

Technical Document

Niagara Edge 10 ACE Driver Guide

niagara

About this guide

This topic contains important information about the purpose, content, context, and intended audience for this document.

Product Documentation

This document is part of the Niagara technical documentation library. Released versions of Niagara software include a complete collection of technical information that is provided in both online help and PDF format. The information in this document is written primarily for Systems Integrators. To make the most of the information in this book, readers should have some training or previous experience with Niagara software, as well as experience working with JACE network controllers.

Document Content

This document describes the initial software installation and configuration of the Edge 10 ACE Driver on a Niagara Edge 10 device, using Workbench (versions Niagara 4.8 and later).

This document does not cover station configuration or Niagara 4 components. For more information on these topics, please refer to online help and various other Niagara 4 software documents.

- [Document change log](#)
Changes to this document are listed in this topic.
- [Related documentation](#)
Additional information on Niagara system, devices and protocols is available in the following documents.

Document change log

Changes to this document are listed in this topic.

February 24, 2020

Edited several topics in the chapter, “ACE Nrio Trunk”.

November 1, 2019

In the topic, “Network architecture”, added a caution note alerting customers to restrict access to all computers, devices, field buses, components, etc., that manage their building model. Added a new chapter on the “ACE Nrio Trunk”.

August 9, 2019

Edited the ace-AceDynamicComp component topic to provide additional usage details on Select objects.

July 19, 2019

Updated content throughout, added two additional component topics to support online help.

March 26, 2019

Minor changes throughout.

February 14, 2019

Early Access documentation

Parent topic: [About this guide](#)

Related documentation

Additional information on Niagara system, devices and protocols is available in the following documents.

- *Niagara Edge 10 Install and Startup Guide*

Parent topic: [About this guide](#)

About the Edge 10 ACE

The Edge 10 ACE Driver is an autonomous control engine that comes with the Niagara 4.8 installation, and runs in parallel with Niagara. The ACE driver provides fast start-up times and deterministic control. ACE's ability to start-up quickly allows an application to quickly assume control over the I/O to perform critical functions. Deterministic control ensures that your application will always run in the specific order and timeframe as specified upon application creation.

The ACE driver runs only on a Niagara Edge 10 device, and is comprised of an ACE network, and a catalog of ACE components. Just like programming a standard Niagara application, you drag ACE components onto a wire sheet which provides a familiar way to create applications. The ACE App is composed of a set of linked components, similar to kitControl components which run in a Niagara station.

The ACE App runs in parallel with a Niagara station and both are able to communicate and share data with each other. While the two engines can collaborate, the application developer must choose if they want to control onboard Edge 10 I/Os and remote I/Os in the ACE App or in the station.

Note: If you try to run both the AceEdgeNetwork and the EdgeIONetwork one or both of the networks will be in fault. If you add the second network it will be in fault. If you restart the station they will both be in fault.

Another feature of the ACE driver is that it is fully compatible with application templates, which are the primary means of sharing solutions that are built for Edge devices. As long as the ACE executable files (*.ace) are in the /ace folder in the station's file space, they will be picked up by any application template that is created from that station and then installed to the same location in the target station. Complete details on application templates are available in the *Niagara Templates Guide*.

- [Requirements](#)
Following are the licensing, software, and platform requirements for running the Edge 10 ACE Driver.
- [ACE Network architecture](#)
The ACE driver uses the standard Niagara network architecture.
- [Installing ACE software](#)
The following procedures describe the steps to install the ACE software on a released device as well as on a pre-released device.
- [Creating an ACE Application offline](#)
This procedure describes the steps to engineer Ace applications offline, for later installation in the field.
- [Adding App logic offline](#)
Initially, the Local AceDevice contains a minimum default application which contains a Services folder with the PlatformService and CommService objects. You drag ACE components to the App Wire Sheet to build your app logic. As an example, this procedure describes adding the Ace Ramp and AoPoint components.
- [Downloading an offline App to the ACE engine](#)
This procedure describes the steps to download (or push) a previously created Ace App (stored offline on your PC file system) to the ACE engine which is running in memory.
- [Creating an ACE application online](#)
On installation, the AceEdgeNetwork comes with a default ACE App. The default App contains only the components necessary for the ACE driver to communicate with the station, it does not contain any logic. This procedure describes the steps to essentially copy and edit the default App which is running in memory.
- [Adding ACE proxy points to the station](#)
The Local ACE Device has a Points extension, which provides a way for the station to interact with the ACE App to gather data. Using **Discovery** you can add ACE proxy points to the station database to transfer data from the App to your station, and to control functions of the App from your station.
- [Troubleshooting tips](#)
This section provides tips to help in troubleshooting your Ace App.

Requirements

Following are the licensing, software, and platform requirements for running the Edge 10 ACE Driver.

The ACE driver is a licensed feature. The platform license must include the “ace” feature.

The following software modules are required to run the ACE driver.

- ace (-rt, -wb)
- aceEdge (-rt)
- platAcelpc (-rt)

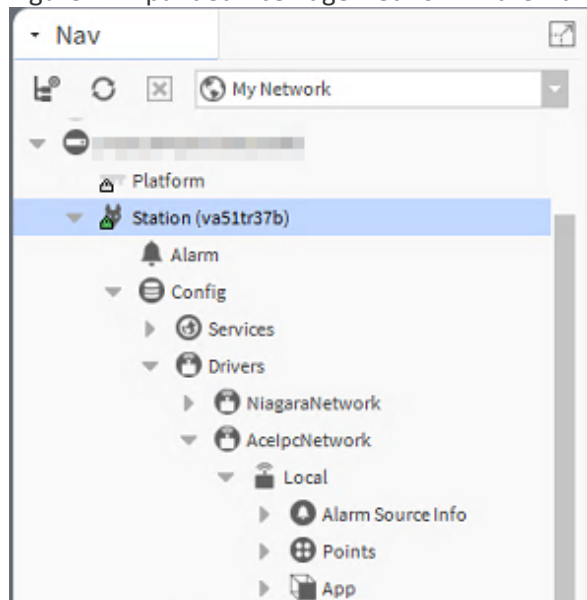
The ACE driver runs on any Niagara Edge 10 platform running Niagara 4.8 or later.

Parent topic: [About the Edge 10 ACE](#)

ACE Network architecture

The ACE driver uses the standard Niagara network architecture.

Figure 1. Expanded Ace Edge Network in the Nav Tree



The AceEdgeNetwork is the top-level parent component, and by default has typical component slots such as, Health and other status properties, Alarm Source Info, Ping Monitor, Tuning Policies, Heart Beat, and etc. The **Property Sheet** is the default view for the AceEdgeNetwork.

The Local (Ace Device) is the second-tier component in the network. It has typical device status properties, a Points Extension, and the ACE App. There can be only one child device in the AceEdgeNetwork, as such the AceDevice is included with the network component. The **Property Sheet** is the default view for the AceDevice.

The AceDevicePointsExt is the parent container for proxy points. The **Ace Points Manager** is the default view for the PointsExt.

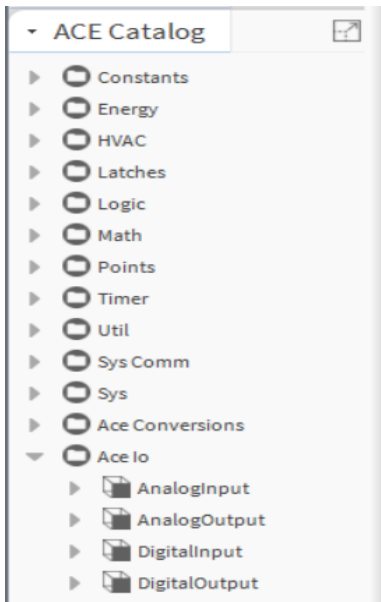
The ACE App is a child of the Local AceDevice. The App folder is a Niagara representation of the ACE application

components. It is useful for visualization of the applications behavior. The proxy points are a sufficient interface to the application. There is no need for the in-station representation of the App.

The ACE App has a number of configurable properties related to scan frequency, scan level, and order. When the ACE application is running it constantly scans components. Setting the scan “Level” is how you configure the frequency of component scanning, while “Order” controls the order of execution within a scan level. For more details see, “ace-AceApp” and “ace-AceCompManager” in the “Components and views” chapter of this guide.

The ACE driver has an extensive catalog of its own ACE components. You drag ACE components onto the **Ace Wire Sheet** view to build the Ace App.

Figure 2. ACE Catalog with expanded Ace Io folder



Note: The **ACE Catalog** functions like a palette, in that you access it via the Workbench Side Bars, in the pane you can navigate to folders, expand folders, and drag components from the pane. The catalog contains a large number of Ace components, many of those are similar to kitControl components.

CAUTION: Protect against unauthorized access by restricting physical access to the computers and devices that manage your building model. Set up user authentication with strong passwords, and secure components by controlling permissions. Failure to observe these recommended precautions could expose your network systems to unauthorized access and tampering.

Parent topic: [About the Edge 10 ACE](#)

Installing ACE software

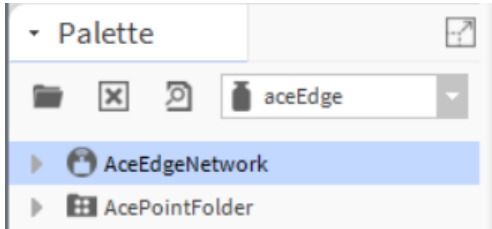
The following procedures describe the steps to install the ACE software on a released device as well as on a pre-released device.

Installing software on a released Edge device

1. Refer to the *Niagara Edge 10 Install and Startup Guide* and follow the instructions on “Setting up a single device using commissioning”, and “Running the Commissioning Wizard”.
2. Connect to the station running on the Edge device.

3. In the station Drivers node, remove the existing EdgeloNetwork
4. Open the aceEdge palette and drag the AceEdgeNetwork to the Drivers node.

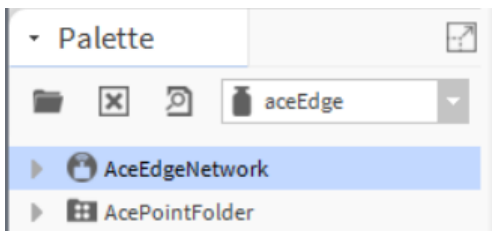
Figure 1. AceEdgeNetwork available in aceEdge palette



Installing on a pre-released Edge device

1. Update the Edge device with the Niagara 4.8 software
2. In the station Drivers node, remove the existing EdgeloNetwork.
3. Open the aceEdge palette and drag the AceEdgeNetwork to the Drivers node.

Figure 2. AceEdgeNetwork available in aceEdge palette



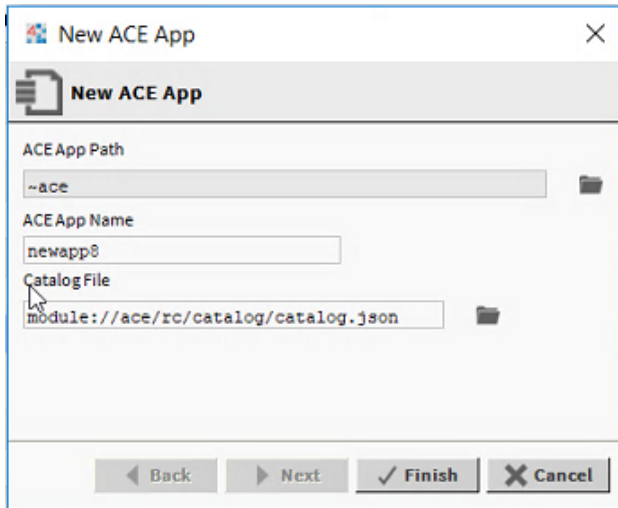
Parent topic: [About the Edge 10 ACE](#)

Creating an ACE Application offline

This procedure describes the steps to engineer Ace applications offline, for later installation in the field.

The PC is properly licensed and running the Niagara 4.8 Workbench.

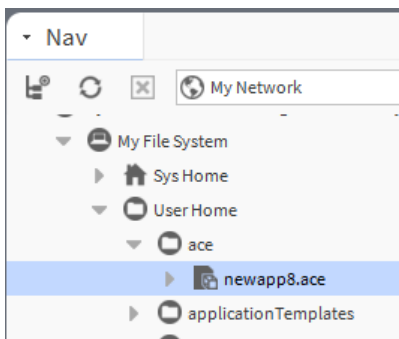
1. In Workbench, click **Tools > New ACE App**.



2. In the **New ACE App** window, click in the ACE App Name field and enter the preferred app name.

Note the ACE App Path which indicates the default location where the new app will be saved (the ~ace folder in your Windows User Home). The path is editable, you can save the App anywhere in your file system. Also note, that there is only one CatalogFile (use the default path shown).

3. Click **Finish**.



The new offline Ace application is created and saved to the ~ace folder in your Windows User Home.

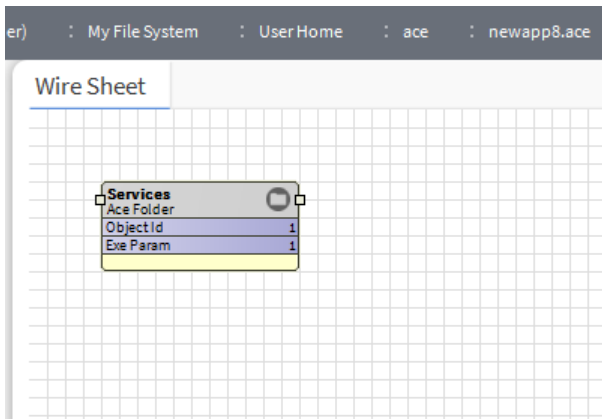
Parent topic: [About the Edge 10 ACE](#)

Adding App logic offline

Initially, the Local AceDevice contains a minimum default application which contains a Services folder with the PlatformService and CommService objects. You drag ACE components to the App Wire Sheet to build your app logic. As an example, this procedure describes adding the Ace Ramp and AoPoint components.

You have already created your new offline App.

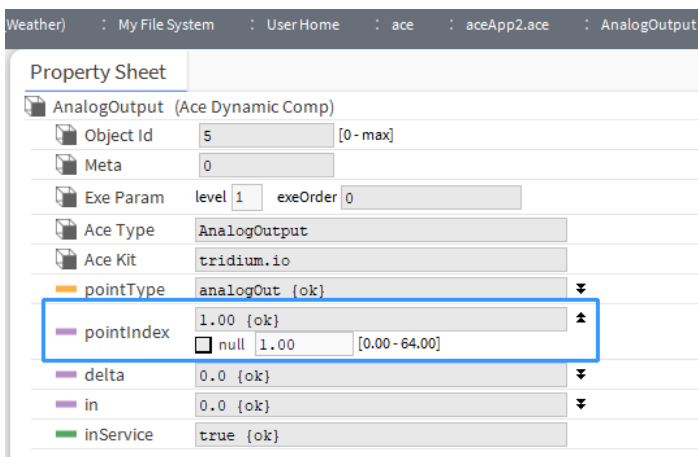
1. In the NavTree, double-click your offline App to open the **Wire Sheet** view.



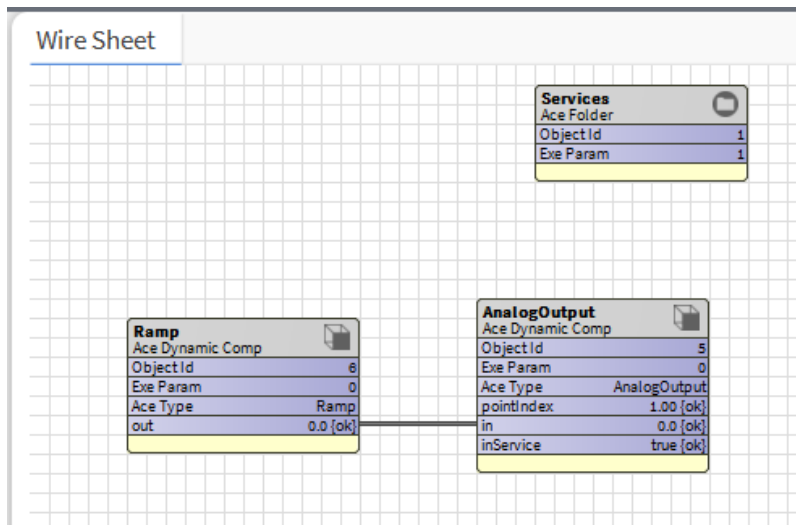
2. In the toolbar, click **Window > Side Bars > ACE Catalog**, to open the **ACE Catalog** pane.

Note: The **ACE Catalog** functions like a palette, in that you access it via the Workbench Side Bars, in the pane you can navigate to folders, expand folders, and drag components from the pane. The catalog contains a large number of Ace components, many of those are similar to **kitControl** components.


3. Expand the catalog's **Util** folder and drag the **Ramp** component to the wire sheet.
4. Expand the **Ace Io** folder and drag the **AnalogOutput** component to the wire sheet.
5. Open a **Property Sheet** view of the AnalogOutput and set the pointIndex value to 1.00 which corresponds to 1st output.



6. Link the output of the Ramp component to the input of the AnalogOutput.



7. Click the **SaveBog** icon in the toolbar to save your changes to the Ace App.

CAUTION: When engineering offline Apps be sure to save your changes via the Workbench toolbar SaveBog icon () not the Save icon. SaveBog saves the App offline.

Parent topic: [About the Edge 10 ACE](#)

Downloading an offline App to the ACE engine

This procedure describes the steps to download (or push) a previously created Ace App (stored offline on your PC file system) to the ACE engine which is running in memory.

- The PC is properly licensed and running the Niagara 4.8 Workbench
 - You have previously created the ACE App offline and saved it to your PC file system.
1. In the station Drivers node, expand the AceEdgeNetwork, right-click **Local** and click **Ace Application...> Download App**.
 2. In the **File Chooser** window, select your previously created offline application and click **OK**.

ACE App download job runs and a confirmation notice displays on-screen when finished. Also, the download job is visible in the JobService.

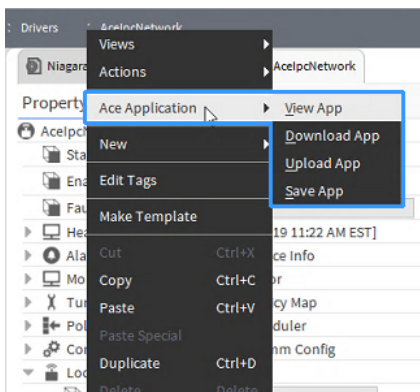
The offline ACE App is pushed to the Ace Device's memory. At this point the App starts running immediately.

Parent topic: [About the Edge 10 ACE](#)

Creating an ACE application online

On installation, the AceEdgeNetwork comes with a default ACE App. The default App contains only the components necessary for the ACE driver to communicate with the station, it does not contain any logic. This procedure describes the steps to essentially copy and edit the default App which is running in memory.

- The PC is properly licensed and running the Niagara 4.8 Workbench
 - The AceEdgeNetwork is already added to the station's Drivers node.
1. In the Drivers node in the NavTree, right-click on the AceEdgeNetwork and open a **Property Sheet** view.
 2. Right-click the **Local** device and in the menu click **Ace Application...> View App**.



This copies the default Ace App (already running in memory) and places it as a child of the Local Ace Device. For more details on the Ace Device right-click menu options see, “ace-AceDevice” in the “Components and views” chapter of this guide.

3. Right-click the **Local** device again and in the menu click **Ace Application > Save App**. This step puts your new App in the ACE engine (it becomes the App running in memory).
4. Double-click on the App to open a **Ace Wire Sheet** view.
5. In the Workbench toolbar, **Window > Side Bars > ACE Catalog**, to open the **ACE Catalog** pane.
6. In the catalog, navigate to desired components and drag them to the wire sheet as needed to build the App logic.
7. In the Workbench toolbar, click the **Save** icon.

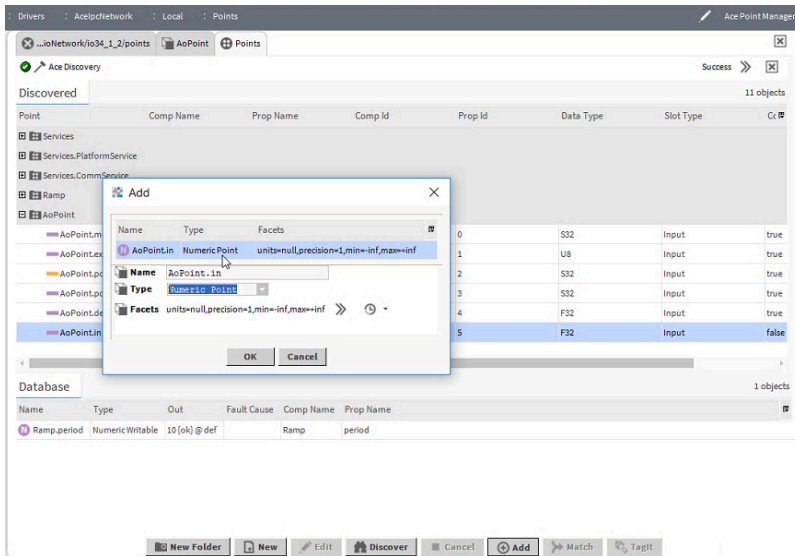
Your new ACE App is saved within the station.

Parent topic: [About the Edge 10 ACE](#)

Adding ACE proxy points to the station

The Local ACE Device has a Points extension, which provides a way for the station to interact with the ACE App to gather data. Using **Discovery** you can add ACE proxy points to the station database to transfer data from the App to your station, and to control functions of the App from your station.

- You have an existing station configured with the AceEdgeNetwork
 - You have an existing ACE App that contains logic already running in memory.
1. Expand the AceEdgeNetwork in the station Drivers node and in the App, double-click the **Points** folder to open the **Ace Point Manager** view.
 2. Click **Discover**.
The **Discover** pane lists one folder for every discovered ACE component in the App.
 3. Click to expand a folder.
This reveals the slots of that component which can be added as proxy points.
 4. Select one or more of the exposed slots and click **Add**.
The selected slots are added as proxy points in the station **Database** pane, as shown in the example image.
 5. In the Workbench toolbar, click the **Save** icon to save your changes to the station.



Parent topic: [About the Edge 10 ACE](#)

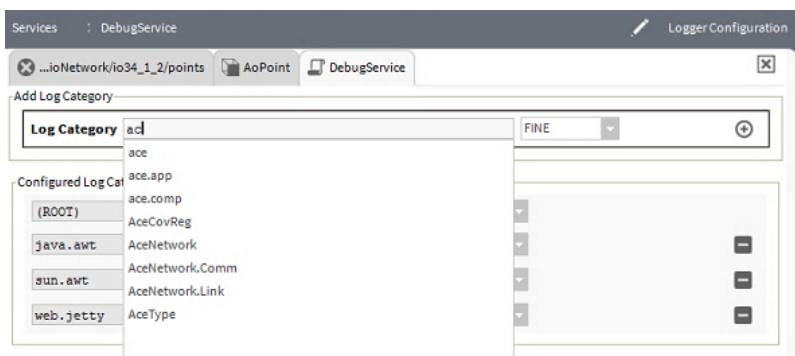
Troubleshooting tips

This section provides tips to help in troubleshooting your Ace App.

Debug options

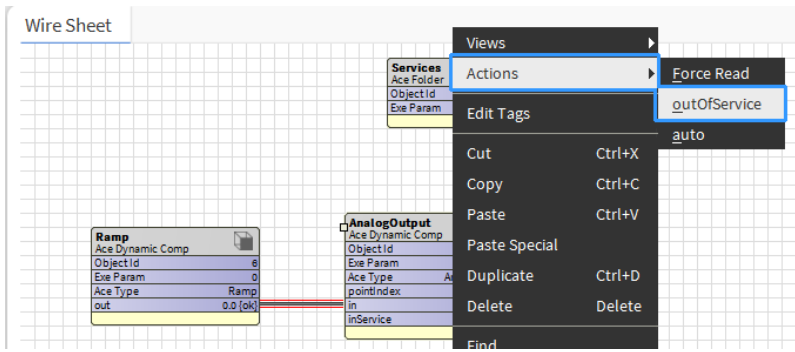
In the station DebugService, there are debug options to add ACE log categories. You can use these to gather ACE App data in the **Application Director**.

Figure 1. Debug Options for ACE log categories



outOfService action

The **outOfService** action is available for all four of the inputs and outputs found in the Ace Io folder of the Ace Catalog. When invoked on an input the **outOfService** action disables the input and sets it to the indicated value. When invoked on an output the **outOfService** action locks the point so that further updates are not written to hardware. Useful when you want to disable certain inputs or outputs while you examine just a portion of the logic. When no longer needed, remove the outOfService setting with the **auto** action.



Parent topic: [About the Edge 10 ACE](#)

ACE Nrio trunk

In Niagara 4.9 and later, the ACE driver provides support for the ACE Nrio Trunk. This allows for Nrio16 and Nrio34 integration with the ACE App via a familiar (Niagara driver) interface.

Under the AceEdgeNetwork, this is enabled via the NrioTrunk Use AceNrio property, which when set to “true” adds a cookie to the station and restarts the ACE engine.

Note: When you have an existing NrioNetwork already enabled and running in the station, that existing NrioNetwork has control of the COM port that the NRIO module is connected to. If you subsequently configure your AceEdgeNetwork to use the ACE Nrio Trunk, the NrioService component in the ACE App is initialized in a fault state because it cannot get a lock on the COM port. The reverse is true as well.

The configuration workflow is as follows.

- In the AceEdgeNetwork, add the Nrio Trunk and enable AceNrio.
- Add the NrioService to the ACE App and configure the communications port.
- Add NrioDevices (Nrio16Device and/or Nrio34Device) as needed to match the physical device(s).
- Add Nrio I/O point components (e.g., NrioAnalogIn, NrioAnalogOut, NrioDigitalIn, NrioDigitalOut) to all read/write access to Nrio points.
- Configure the pulse counter.
- [NrioService](#)
NrioService is an implementation of the ace-AceDynamicComp component. It must be present (on the ACE Edge network) for other ACE Nrio components to function.
- [aceEdge-NrioDevice](#)
NrioDevice (an implementation of the ace-AceDynamicComp component) represents a specific device. Map Nrio16Device and Nrio34Device components to the corresponding physical IO-R modules. Configure properties to address the device.
- [aceEdge-ioPoints](#)
The Nrio I/O Points read or write data to a single point.
- [Setting up the Nrio Trunk](#)
This procedure describes how to install and configure the Nrio Trunk on the ACE driver.
- [Setting up the NrioService](#)
This procedure describes how to set up the NrioService in the ACE App to use the ACE Nrio Points components to control actual IO-R modules.
- [Adding Nrio Points](#)
This procedure describes how to add and configure the NrioPoints in a NrioDevice.

NrioService

NrioService is an implementation of the ace-AceDynamicComp component. It must be present (on the ACE Edge network) for other ACE Nrio components to function.

Properties

This component is found in the **ACE Catalog**, AceNrio folder.

Figure 1. NrioService properties

NrioService (Ace Dynamic Comp)

Object Id	7	[0 - max]
Meta	201924864	
Exe Param	level 1	exeOrder 0
Ace Type	NrioService	
Ace Kit	tridium.nrio	
comPort	/dev/ser1	
trunk	1	
baudRate	115200	
enable	<input checked="" type="checkbox"/> true	
fault	<input type="checkbox"/> false	
faultCause	OK	
commLossTimeout	8	s [8 - 900]
startupTimeout	15	s [8 - 900]
logLevel	Warning	

In addition to standard health properties and the properties common to all instances of the Ace Dynamic Comp component, the following configuration properties are present.

Name	Value	Description
comPort	/dev/ser1 (default)	
trunk	1	
baudRate	115200	Communications transmission rate
enable	true, false (default)	
commLossTimeout	8 (default)	Number of seconds that may pass before timing-out on loss of comm connections. Range is 8–900.
startupTimeout	15 (default)	Number of seconds that may pass before timing-out on startup. Range is 8–900.

Parent topic: [ACE Nrio trunk](#)

aceEdge-NrioDevice

NrioDevice (an implementation of the ace-AceDynamicComp component) represents a specific device. Map Nrio16Device and Nrio34Device components to the corresponding physical IO-R modules. Configure properties to address the device.

You can map the NrioDevice component to a specific IO-R module via the **Match** window.

This component is found in the **ACE Catalog**, AceNrio folder.

Figure 1. Nrio34Device properties (same for Nrio16Device)

Nrio34Device (Ace Dynamic Comp)	
Object Id	6 [0 - max]
Meta	202248448
Exe Param	level 1 exeOrder 0
Ace Type	Nrio34Device
Ace Kit	tridium.nrio
status	{ok}
uid	00001958A013
devAddress	3
enableCommLossDefaults	<input type="radio"/> false
enableStartupDefaults	<input type="radio"/> false
commLossTimeout	8 s
startupTimeout	15 s
devVersion	2.1;2.1
▶ NrioDigitalOutput	Ace Dynamic Comp

In addition to common AceDynamicComp and health properties, the following configurable properties are present.

Name	Value	Description
devAddress		
enableCommLossDefaults	true, false (default)	
enableStartupDefaults	true, false (default)	
commLossTimeout	8 (default)	
startupTimeout	15 (default)	
devVersion		

Actions

Following are right-click menu options for Nrio16Device and Nrio34Device components

- **Force Read** —
- **Wink** — toggles the digital output (DO1) of the selected device. This action is enabled when the device status is "OK".
- **Ping** — checks on whether the NrioDevice is in service. This action is enabled when the device is matched.
- **Match** — opens the **Match** window, triggering the match and discovering process for the NrioDeivce. This action is enabled when the device status is not "OK".

Parent topic: [ACE Nrio trunk](#)

aceEdge-ioPoints

The Nrio I/O Points read or write data to a single point.

NrioPoints are implementations of the AceDynamicComp component.

- NrioAnalogInput
- NrioAnalogOutput
- NrioDigitalInput
- NrioDigitalOutput

All Nrio I/O point components must have an NrioDevice parent to obtain an address.

This component is found in the **ACE Catalog**, AceNrio folder.

Figure 1. NrioDigitalOutput properties

Property	Value	Range/Options
Object Id	9	[0 - max]
Meta	201790208	
Exe Param	level 1	exeOrder 0
Ace Type	NrioDigitalOutput	
Ace Kit	tridium.nrio	
status	{ok}	
pointIndex	1	[1 - 64]
polarity	normal	
minOn	0	ms [0 - max]
minOff	0	ms [0 - max]
defaultValue	false	
in	false	
out	false {ok}	

In addition to common AceDynamicComp and health properties the following configurable properties are present.

Name	Value	Description
pointIndex		Configure the Point Index property to map the component to a certain I/O. If the value of Point Index property is invalid (duplicated or out of valid range), the component will be in fault.
polarity		
minOn		
minOff		

Name	Value	Description
defaultValue		
in		
out		

Parent topic: [ACE Nrio trunk](#)

Setting up the Nrio Trunk

This procedure describes how to install and configure the Nrio Trunk on the ACE driver.

- You are connected to the station running on an Edge device.
 - There is an existing AceEdgeNetwork with ACE App installed and running.
 - The aceEdge palette is open.
- Under the station's Drivers node, open a **Property Sheet** view of the AceEdgeNetwork.
 - From the aceEdge palette, drop the NrioTrunk component onto the network.
 - Open a **Property Sheet** view of the NrioTrunk and set Use Ace Nrio to "true".

Note: Leave the default value of Trunk and Port Name properties unchanged.

The ACE App is configured for Nrio.

Parent topic: [ACE Nrio trunk](#)

Setting up the NrioService

This procedure describes how to set up the NrioService in the ACE App to use the ACE Nrio Points components to control actual IO-R modules.

- You are connected to the station running on an Edge device.
 - There is an existing AceEdgeNetwork with ACE App installed and running.
 - The ACE driver is configured to use AceNrio.
 - The aceEdge palette is open.
- Open the ACE App wire sheet.
 - Open the **ACE Catalog** sidebar.
 - From the Ace Nrio subfolder in the sidebar, drag the NrioService onto the App wire sheet.

Note: The NrioService is required in the ACE app in order to use NRIO devices.

- From the Ace Nrio subfolder in the sidebar, drop an NrioXXDevice onto the ACE App wiresheet, where NrioXXDevice is either Nrio16Device or Nrio34Device.
- Right-click the NrioDevice and click **Actions > Match**.
- In the Match window, click **Discover**, click to select one of the discovered UUIDs and click **Match**. The NrioDevice component's status should be OK

The ACE App is configured for Nrio. You can add points next.

Parent topic: [ACE Nrio trunk](#)

Adding Nrio Points

This procedure describes how to add and configure the NrioPoints in a NrioDevice.

Note: Nrio I/O point components are required to reside in a certain NrioDevice component, i.e. they must have a NrioDevice as their parent component.

Once added, you need to configure the NrioPoint's Point Index property to map the component to a certain I/O. If the value of Point Index property is invalid (duplicated or out of valid range), the status of that component will be in fault.

1. Open a **Property Sheet** (or **Wire Sheet**) view of the NrioDevice.
2. From the **ACE Catalog**, drag a point of the desired type onto the view.
3. On the **Property Sheet** for the point, configure the following properties:
 - For Point Index **enter the number of the I/O (e.g., for a01, you enter "1")**.
 - For Inputs, **select the input type (Resistance or Voltage)**.

Parent topic: [ACE Nrio trunk](#)

Components

Components include services, folders and other model building blocks associated with a module. You drag them to a property or wire sheet from a palette. Views are plugins that can be accessed by double-clicking a component in the Nav tree or right-clicking a component and selecting its view from the **Views** menu.

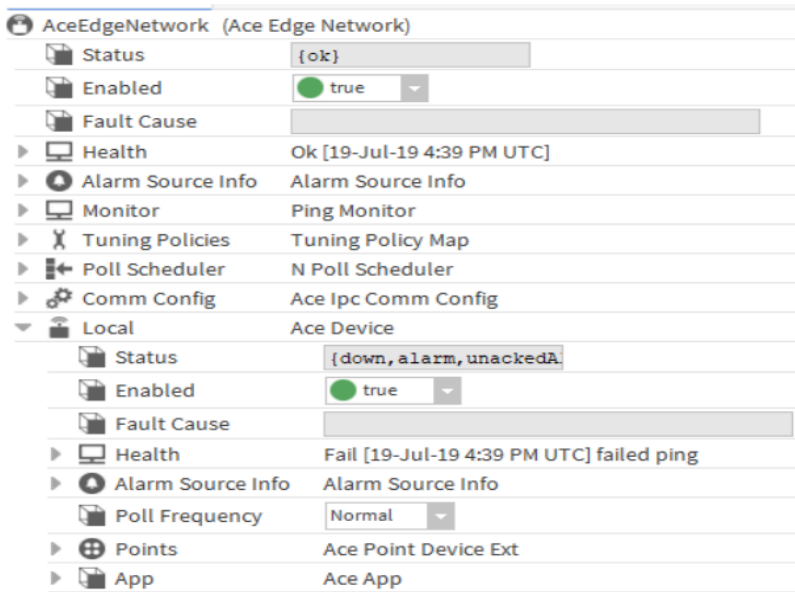
The component and view topics that follow appear as context-sensitive help topics when accessed by:

- Right-clicking on the object and selecting **Views > Guide Help**
- Clicking **Help > Guide On Target**
- [aceEdge-AceEdgeNetwork](#)
The AceEdgeNetwork component is added to the station Drivers node. The network includes a single Local AceDevice. Also, there is no **Device Manager** view for this network since there can be only one device. Double-clicking the AceEdgeNetwork opens a **Property Sheet** view. The component is found in the aceEdge palette.
- [ace-AceDevice](#)
By default, the Local (Ace Device) object is included in the AceEdgeNetwork.
- [ace-AcePointDeviceExt](#)
The **Ace Point Manager** is the default view of the Points Ext. The Discover action lists one folder for every discovered component in the Ace App. Expanding a folder reveals the properties of that component. The properties can be added as proxy points in the station database.
- [ace-Ace App](#)
The ACE App runs in memory.
- [ace-AceFolder](#)
By default, each ACE App contains the “Services” Ace Folder, a container for additional ACE objects. By default, this folder contains the PlatformServices and the CommService components. You can add additional AceFolders for your own purposes.
- [ace-AcePointFolder](#)
- [ace-AceDynamicComp](#)
The AceDynamicComp, a hidden component, is a representation of a runtime ACE component containing a dynamically created list of properties.
- [ace-AceCompManager](#)
The **Ace Comp Manager** view is available from the **Views** dropdown list in the **ACE Wire Sheet** view (or in the **NavTree**, from the right-click menu for the App or for any Folder (component) in the App). Use this view to compare and edit the Level and Order settings for components in the App.
- [ace-AcePointManager](#)
The default view of the AceDevicePointExt. Like other manager views, it provides the Discover, Add, and etc., actions.
- [ace-AceWireSheet](#)
The default view of the AceApp and any Folders in the App. Like the Niagara **Wire Sheet** view, the **Ace Wire Sheet** provides a familiar space for creating applications.

aceEdge-AceEdgeNetwork

The AceEdgeNetwork component is added to the station Drivers node. The network includes a single Local AceDevice. Also, there is no **Device Manager** view for this network since there can be only one device. Double-clicking the AceEdgeNetwork opens a **Property Sheet** view. The component is found in the aceEdge palette.

Figure 1. AceEdgeNetwork Property Sheet view



In addition to the standard network properties (Status, Enabled, Fault Cause, Health, etc.) the following configurable properties are present.

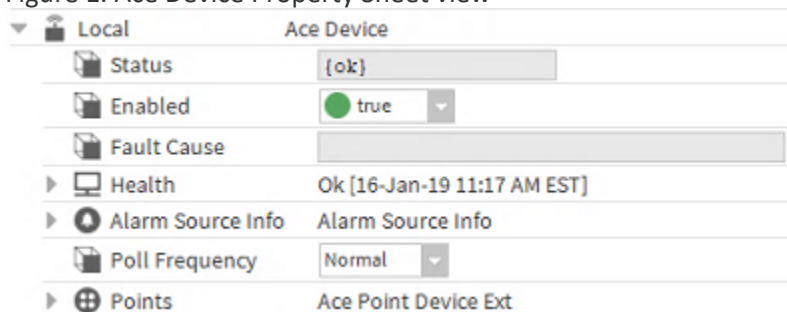
Type	Value	Description
Comm Config		Contains the Fault Cause subproperty

Parent topic: [Components](#)

ace-AceDevice

By default, the Local (Ace Device) object is included in the AceEdgeNetwork.

Figure 1. Ace Device Property Sheet view



The Ace Device contains standard device properties including a Points Ext. Complete details on these standard properties are available in the Niagara Drivers Guide.

Actions

Right-clicking **Local > Ace Application** invokes a menu of application-related actions. The actions function as described here:

- **View App** — transfers the app currently running in memory, putting a copy into the station.

- **Download App** — transfers the app from the hard drive (offline) to the ACE engine (running in memory).
- **Upload App** — transfers (or pushes) the app from ACE engine (running in memory) to the hard drive (offline).
- **Save App** — transfers the app that is currently in the station and puts it into memory (overwriting the app that is running there).

Parent topic: [Components](#)

ace-AcePointDeviceExt

The **Ace Point Manager** is the default view of the Points Ext. The Discover action lists one folder for every discovered component in the Ace App. Expanding a folder reveals the properties of that component. The properties can be added as proxy points in the station database.

The Property Sheet view of the AcePointDeviceExt includes the following configurable Ace Point Discovery Preferences.

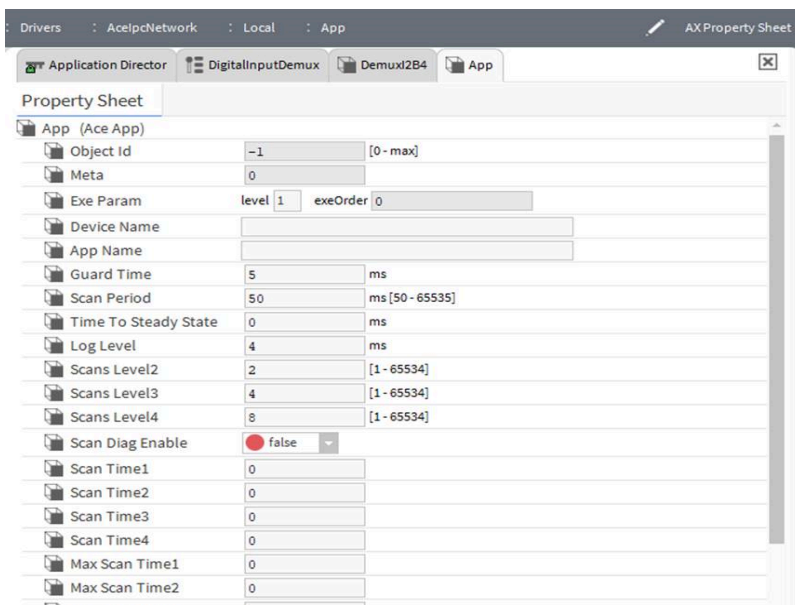
Name	Value	Description
Do Not Ask Again	true, (default) false	
Learn Offline	true, false (default)	

Parent topic: [Components](#)

ace-Ace App

The ACE App runs in memory.

Figure 1. Ace App properties



When the Ace application is running it constantly scans the components. The scan frequency is configurable. Setting the Scan "Level" is how you configure the frequency of component scanning. See the Scan Period

property (50ms default) in the App property sheet view.

- Scan Level 1 (not visible) uses the configured Scan Period rate.
- Scan Levels 2-4 (values are 2, 4, 8 default), where the specified value is a multiplier of the configured Scan Period value.

For example, Level 2 has a value of 2 which means it will scan every 100ms (2 x 50ms). This is useful if you have components that you want to have operate very quickly to be responsive.

Name	Value	Description
Object Id		
Meta		
Device Name		
App Name		
Guard Time		
Scan Period	50 ms (default)	Sets the scan frequency. This value is the basis for the Scans Level 1–4. Scan Level 1 always uses this scan frequency.
Time To Steady State		
Log Level		
Scans Level 1 (not visible)	1	Uses the configured Scan Period value (50 ms, by default).
Scans Level 2	2 (default)	Uses the configured Scan Period value and applies this multiplier to determine the scan frequency. For example, Scans Level 2 scans every 100 ms (2 x 50 ms). Range is 1–65534.
Scans Level 3	4 (default)	Uses the configured Scan Period value and applies this multiplier to determine the scan frequency. Range is 1–65534.
Scans Level 4	8 (default)	Uses the configured Scan Period value and applies this multiplier to determine the scan frequency. Range is 1–65534.

Parent topic: [Components](#)

ace-AceFolder

By default, each ACE App contains the “Services” Ace Folder, a container for additional ACE objects. By default, this folder contains the PlatformServices and the CommService components. You can add additional AceFolders for your own purposes.

Name	Value	Description
Object Id	numeric	A numeric object identifier. The range is 0-65534.
Exe Param	numeric	Executable parameter lists the executable scan level and scan order for this component.

Parent topic: [Components](#)

ace-AcePointFolder

The AcePointFolder is provided for organizing proxy points, if desired. This component is found in the aceEdge palette.

Parent topic: [Components](#)

ace-AceDynamicComp

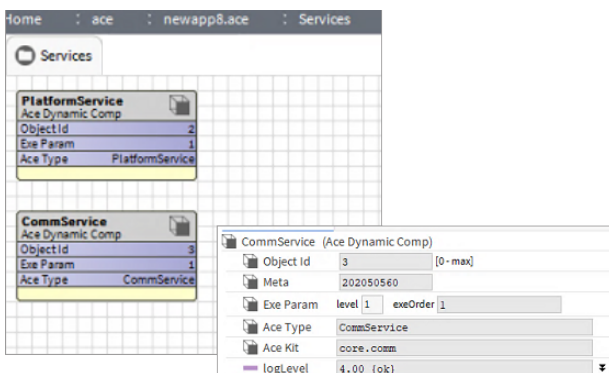
The AceDynamicComp, a hidden component, is a representation of a runtime ACE component containing a dynamically created list of properties.

Common properties

Most of the components in the ACE Catalog are AceDynamicComp components which are created dynamically from code unique to the Niagara Edge 10 platform and included in the ACE driver. Click the following link for a complete list of these components and their properties.

AceDynamicComp Index

Figure 1. Common properties on AceDynamicComp objects



Name	Value	Description
Object Id	2 (default)	This is the unique object identifier that is automatically generated by ACE. It is used by the Niagara Proxy Point to identify which component the proxy point is related to. The range is 0-65534.
Meta	202050560 (default)	This is the unique data that identifies what the functionality of the AceDynamicComp will be.

Name	Value	Description
Exe Param	numeric, 1 (default)	Executable parameter lists the executable scan level and scan order for this component.
Ace Type	text string	Name of this Ace object type
Ace Kit	text string	Container in the ACE Catalog where this component may be found.
logLevel	text string	Shows the configured log level as either Error, Warning, Message, or Trace.

Select components usage details

When working with ACE select components (from the Ace Catalog Select folder), the Select slot requires an Integer value (or a StatusInt slot type) as a link. Many components in the ACE Catalog have numeric out slots, which are of type StatusDouble. These slots must be converted to a StatusInt to be compatible with the Select object Select slot when linking into the Select object. The NumericToInt conversion object must be used as shown below to accomplish this conversion. The NumericToInt component is found in the Conversion folder.

When trying to link an out slot of an ACE object to the select slot of an ACE Select object, and the link will not connect, you can use the following steps to verify that the two slots are not compatible.

1. Put the mouse on the out slot of the "Link From" object and click the mouse.
2. While holding the mouse button, move to the white link bar at the bottom of the ACE Select object to see the link error view.
3. The out slot of the source object should be highlighted. Hover your mouse over the select slot of the target (Select) object and the error message appears, as shown below. This error is a definite indication that the NumericToInt conversion is required.

Figure 2. Example Select object link error

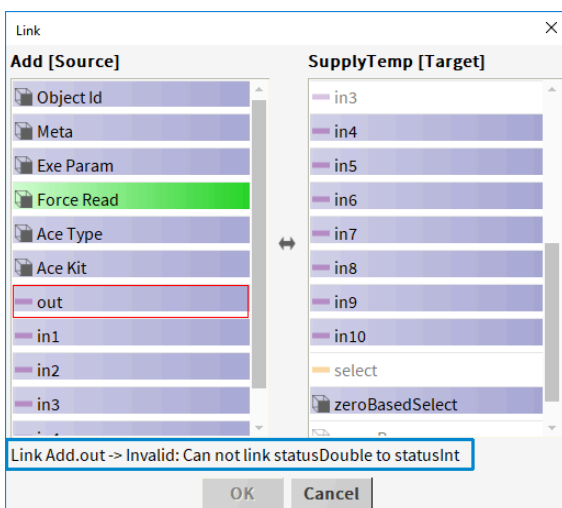
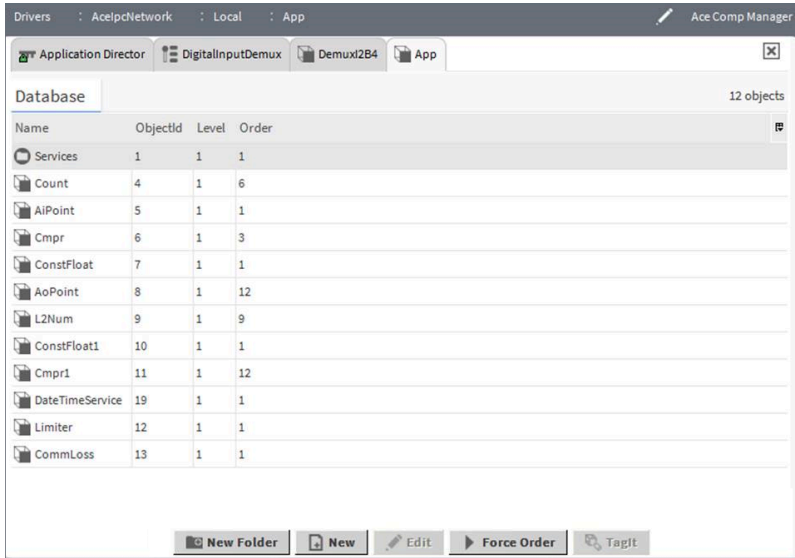


Figure 3. Example usage of NumericToInt conversion component

Property Sheet

Figure 1. Ace Comp Manager view shows components in the Ace App



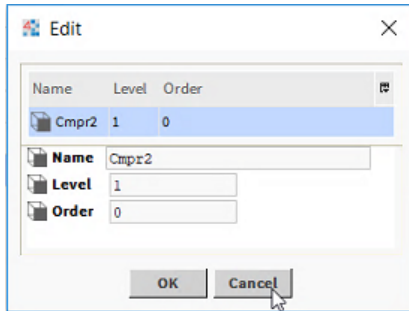
When the ACE App is running it constantly scans the components. The scan frequency is configurable via Level and Order settings. Scan “Level” is how you configure the frequency of component scanning. See Scan Period property (50 ms default) in the ACE App **Property Sheet** view. Scan Level 1 (not visible) uses the configured Scan Period rate. For Scan Levels 2-4 (values are 2, 4, 8 by default), each value is a multiplier of the configured Scan Period value. For example, Level 2 has a value of 2 which means it will scan every 100ms (2 x 50ms). Useful if you have components that you want to have operate really quickly to be responsive.

Order is the order of execution within a scan level. For example, you can drag 3 Ace Compare components onto the wire sheet and link them (1st to the 2nd, the 2nd to the 3rd) to create a logic chain. Save those changes and switch to the **Ace Comp Manager** view. You will see 0 as the Order values of those components. But, you can click **Edit** and set the Level and Order manually, or set order by clicking the **Force Order** button to evaluate the components of the App and determine an execution order based on that.

Name	Value	Description
Level	numeric	Level sets the frequency of component scanning.
Order	numeric	Order sets the order of execution within a scan level.

Buttons

- **New Folder** — creates a new component folder
- **New** —
- **Edit** — opens the Edit window on the selected component where you can modify the Name, Level, and Order settings.



- Force Order — evaluates the components of the App and determines an execution order based on that.
- TagIt —

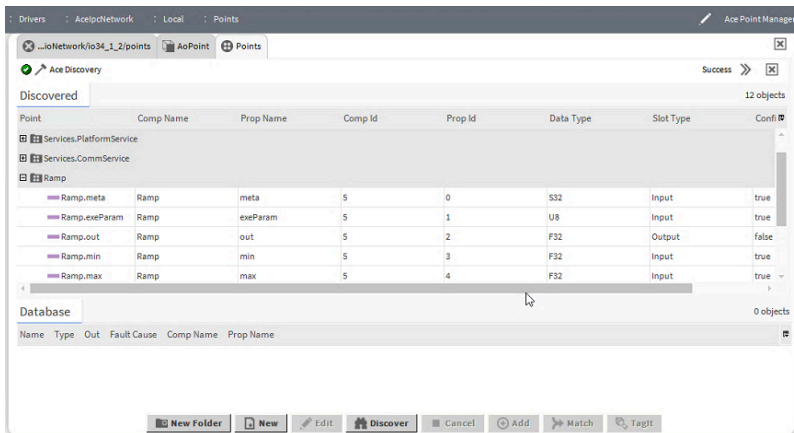
Parent topic: [Components](#)

ace-AcePointManager

The default view of the AceDevicePointExt. Like other manager views, it provides the Discover, Add, and etc., actions.

Discover provides a method of getting data from the ACE App to your station, and enables you to control functions of the App from your station. Discover shows one folder for every discovered component in the ACE App. Expanding one reveals the slots of that component. The slots can be added as proxy points in the station

Figure 1. Ace Point Manager view showing expanded Ramp component properties

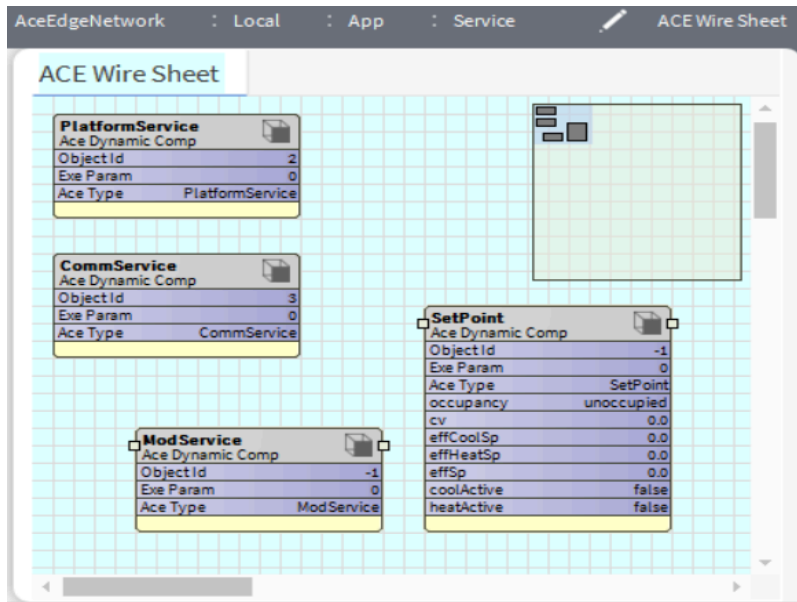


Parent topic: [Components](#)

ace-AceWireSheet

The default view of the AceApp and any Folders in the App. Like the Niagara **Wire Sheet** view, the **Ace Wire Sheet** provides a familiar space for creating applications.

Figure 1. ACE Wire Sheet view of the Service folder in the AceApp



Parent topic: [Components](#)