Technical Document

Niagara Edge 10 ACE Driver Guide

February 24, 2020

Niagara Edge 10 ACE Driver Guide

Tridium, Inc.
3951 Westerre Parkway, Suite 350
Richmond, Virginia 23233
U.S.A.

Confidentiality

Trademark notice

Copyright and patent notice

Contents

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.

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

Related documentation

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

- *Niagara Edge 10 Install and Startup Guide*

Chapter 1 About the Edge 10 ACE

Topics covered in this chapter

- ♦ Requirements
- ♦ ACE Network architecture
- ♦ Installing ACE software
- ♦ Creating an ACE Application offline
- ♦ Adding App logic offline
- ♦ Downloading an offline App to the ACE engine
- ♦ Creating an ACE application online
- ♦ Adding ACE proxy points to the station
- ♦ Troubleshooting tips

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.

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.

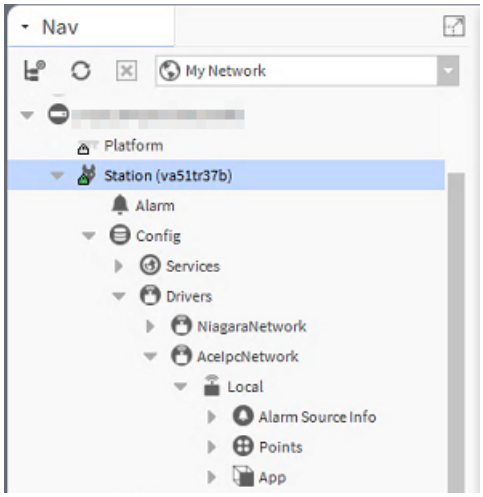
- ace (-rt, -wb)
- aceEdge (-rt)
- platAceIpc (-rt)

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

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.

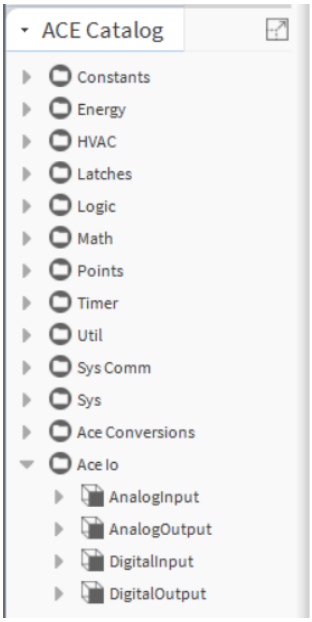
The AceDevicePointsExt is the parent container for proxy points. The **Ace Points Manager** is the default view for the PointsExt.

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 application's 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.

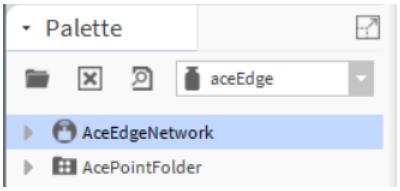
Installing ACE software

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 EdgeIoNetwork
4. Open the **aceEdge** palette and drag the AceEdgeNetwork to the Drivers node.

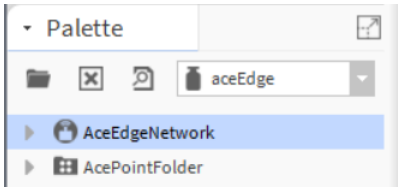
Figure 3 AceEdgeNetwork available in aceEdge palette



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 EdgeIoNetwork.
3. Open the **aceEdge** palette and drag the AceEdgeNetwork to the Drivers node.

Figure 4 AceEdgeNetwork available in aceEdge palette



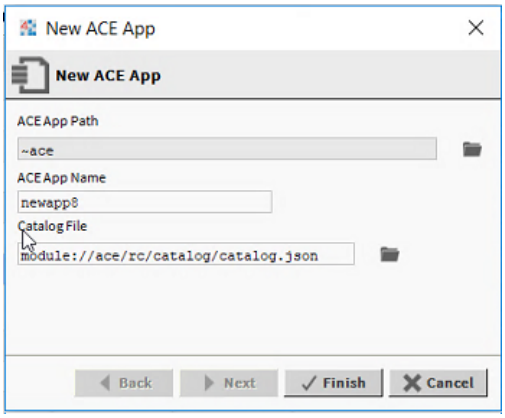
Creating an ACE Application offline

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

Prerequisites:

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

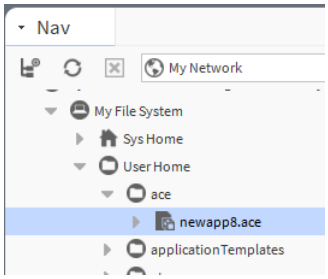
Step 1 In Workbench, click **Tools→New ACE App**.



Step 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).

Step 3 Click **Finish**.



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

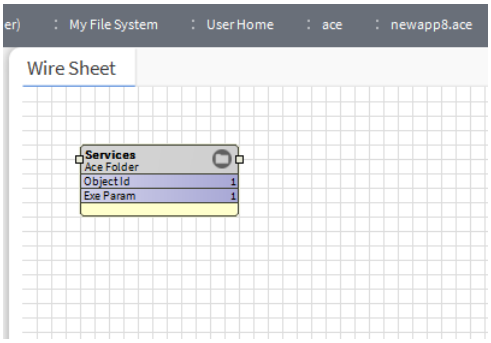
Adding App logic offline

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.

Prerequisites:

You have already created your new offline App.

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



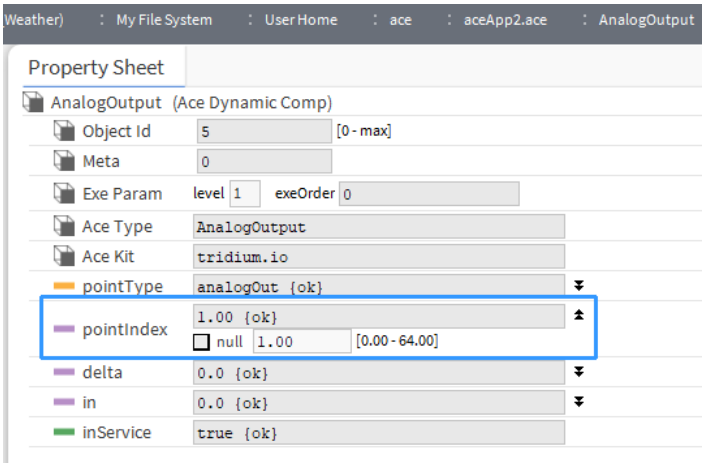
Step 2 In the toolbar, click **Window→Side Bars→ACE Catalog**, to open the **ACE Catalog** pane.

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.

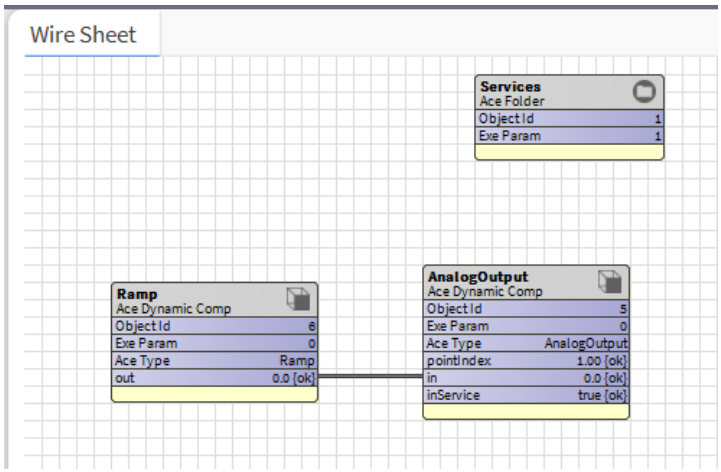
Step 3 Expand the catalog's **Util** folder and drag the **Ramp** component to the wire sheet.

Step 4 Expand the **Ace Io** folder and drag the **AnalogOutput** component to the wire sheet.

Step 5 Open a **Property Sheet** view of the AnalogOutput and set the **`pointIndex`** value to 1.00 which corresponds to 1st output.



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



Step 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 tool-bar **SaveBog** icon ( ) not the Save icon. SaveBog saves the App offline.

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.

Prerequisites:

- 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.

Step 1 In the station Drivers node, expand the AceEdgeNetwork, right-click **Local** and click **Ace Application→Download App**.

Step 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.

Creating an ACE application online

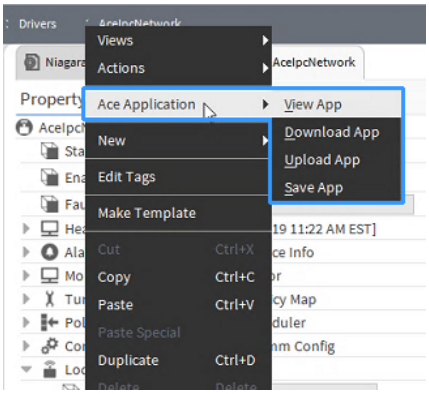
On installation, the AceEdgeNetwork comes with a default ACE App. The default App contains only the components necessary for the ACE driver to communicate with the station, it does not contain any logic. This procedure describes the steps to essentially copy and edit the default App which is running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

Prerequisites:

- The PC is properly licensed and running the Niagara 4.8 Workbench
- The AceEdgeNetwork is already added to the station's Drivers node.

Step 1 In the Drivers node in the NavTree, right-click on the AceEdgeNetwork and open a **Property Sheet** view.

Step 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.

Step 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).

Step 4 Double-click on the App to open a **Ace Wire Sheet** view.

Step 5 In the Workbench toolbar, **Window→Side Bars→ACE Catalog**, to open the **ACE Catalog** pane.

Step 6 In the catalog, navigate to desired components and drag them to the wire sheet as needed to build the App logic.

Step 7 In the Workbench toolbar, click the **Save** icon.

Your new ACE App is saved within the station.

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.

Prerequisites:

- You have an existing station configured with the AceEdgeNetwork
- You have an existing ACE App that contains logic already running in memory.

Step 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.

Step 2 Click **Discover**.

The **Discover** pane lists one folder for every discovered ACE component in the App.

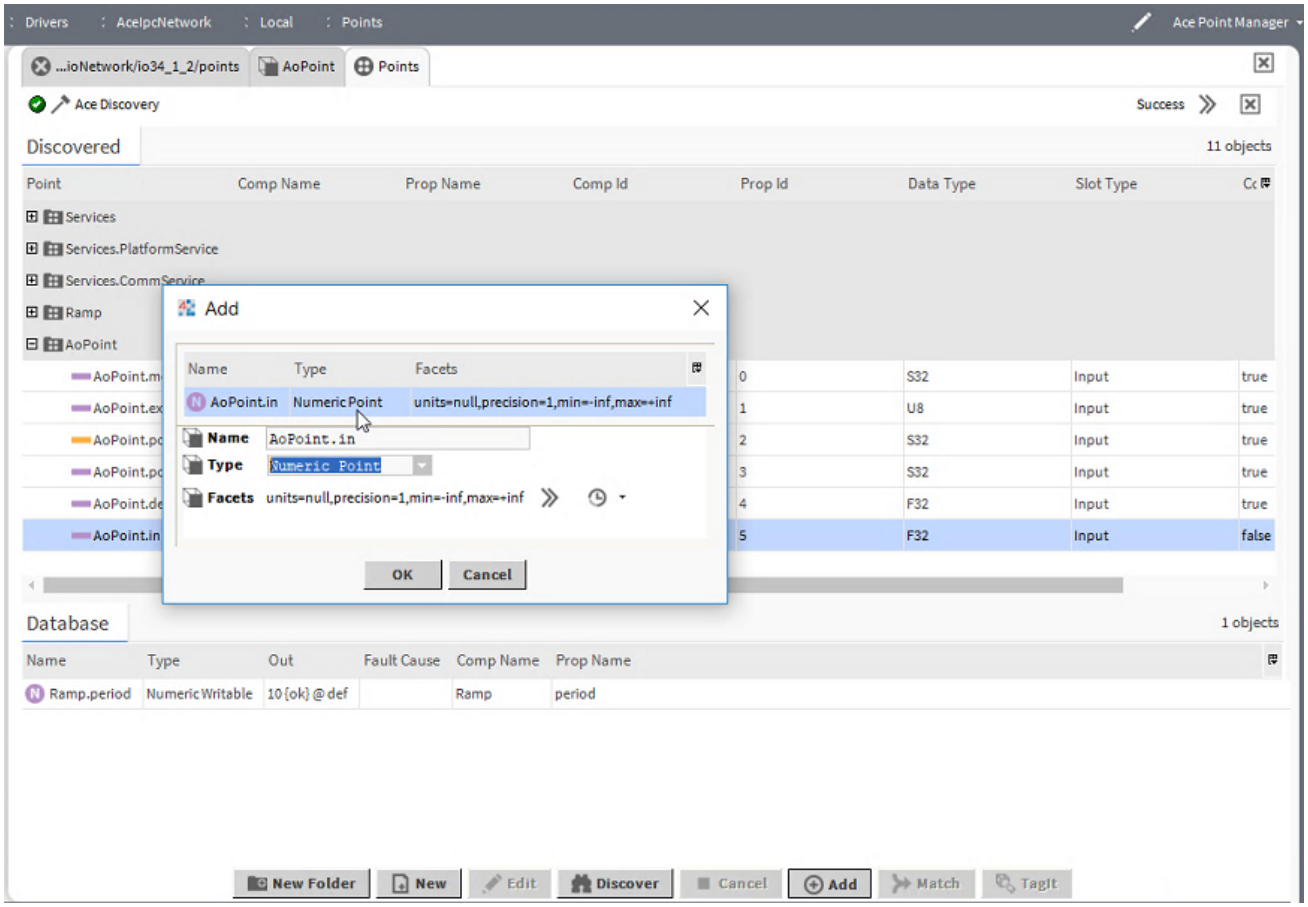
Step 3 Click to expand a folder.

This reveals the slots of that component which can be added as proxy points.

Step 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.

Step 5 In the Workbench toolbar, click the **Save** icon to save your changes to the station.



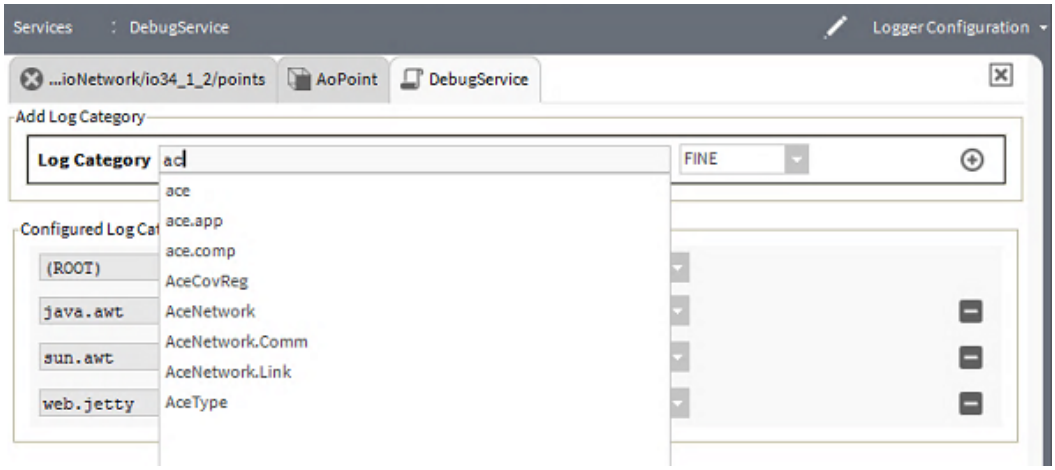
Troubleshooting tips

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

Debug options

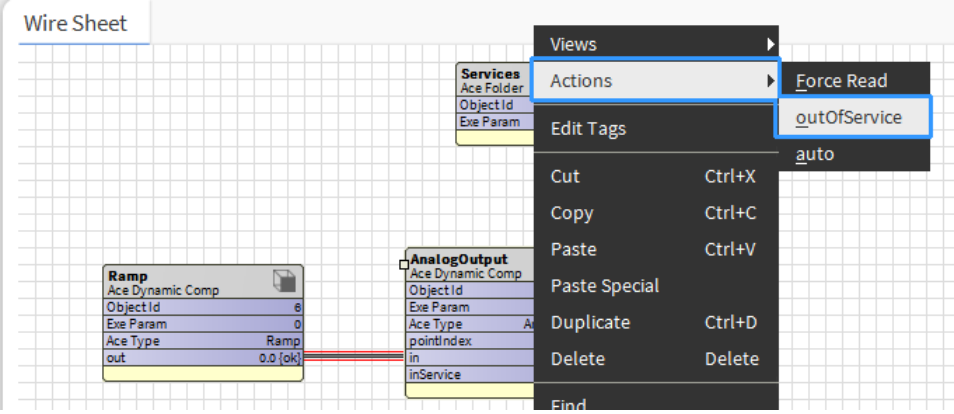
In the station DebugService, there are debug options to add ACE log categories. You can use these to gather ACE App data in the **Application Director**.

Figure 5 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.



Chapter 2 ACE Nrio trunk

Topics covered in this chapter

- ♦ NrioService
- ♦ aceEdge-NrioDevice
- ♦ aceEdge-ioPoints
- ♦ Setting up the Nrio Trunk
- ♦ Setting up the NrioService
- ♦ Adding Nrio Points

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 Niro 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.

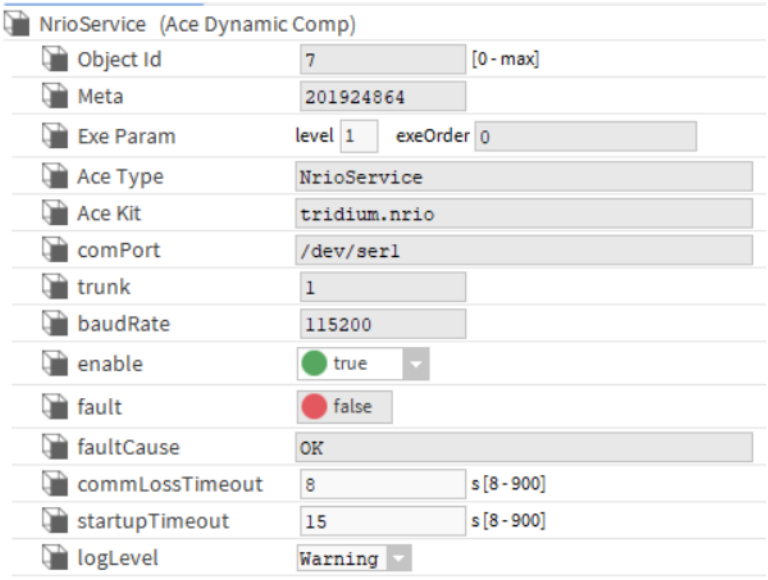
- In the AceEdgeNetwork, add the Nrio Trunk and enable AceNrio.
- Add the NrioService to the ACE App and configure the communications port.
- Add NrioDevices (Nrio16Device and/or Nrio34Device) as needed to match the physical device(s).
- Add Nrio I/O point components (e.g., NrioAnalogIn, NrioAnalogOut, 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 App wiresheet) for other ACE Nrio components to function.

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

Figure 6 NrioService properties



Properties

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 start-up. Range is 8–900.

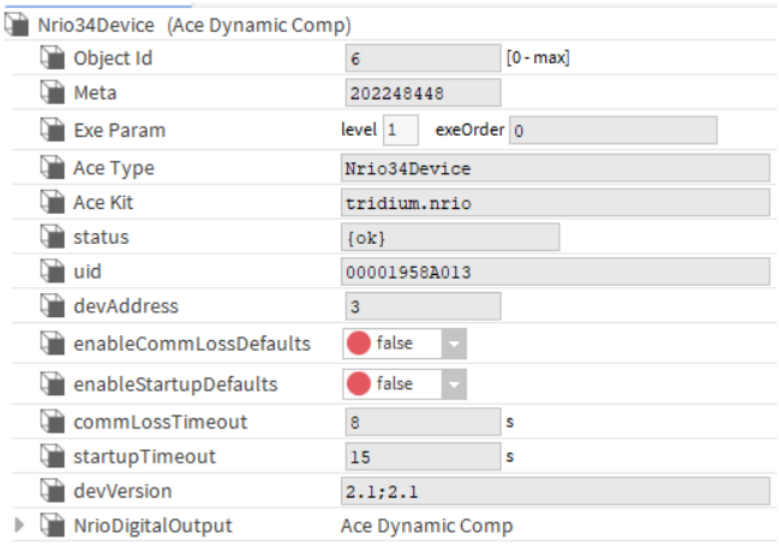
aceEdge-NrioDevice

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 7 Nrio34Device properties (same for Nrio16Device)



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

Name	Value	Description
devAddress		
enableCommLoss-Defaults	true, false (default)	
enableStartupDe-faults	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".

aceEdge-ioPoints

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

NrioPoints are implementations of the AceDynamicComp component.

- NrioAnalogInput
- NrioAnalogOutput

Ace Dynamic Comp Index

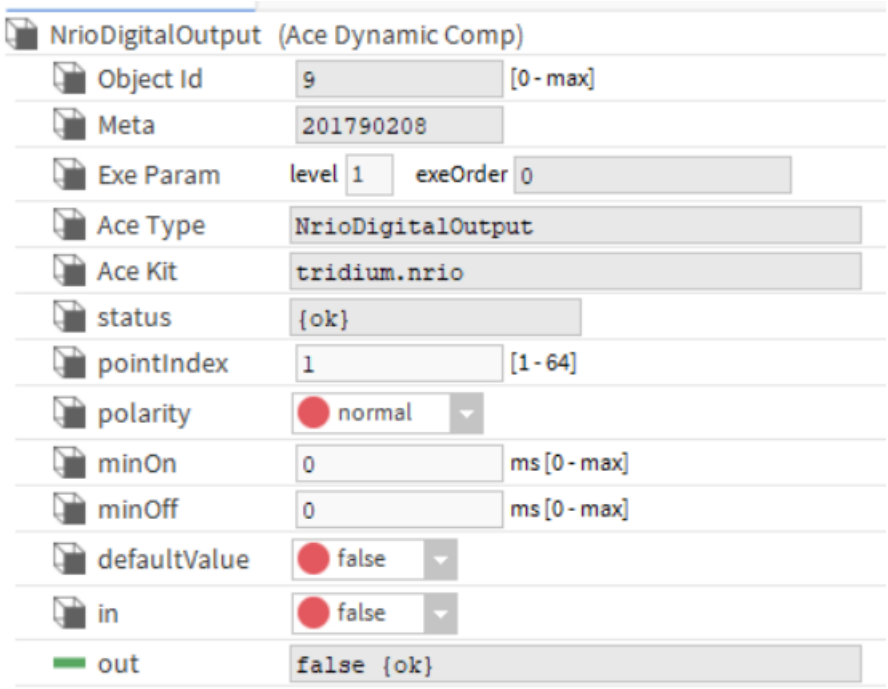
Data Structure Index

- 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 8 NrioDigitalOutput properties



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		
defaultValue		
in		
out		

Setting up the Nrio Trunk

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

Prerequisites:

- 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.

Step 1 Under the station's Drivers node, open a **Property Sheet** view of the AceEdgeNetwork.

Step 2 From the **aceEdge** palette, drop the NrioTrunk component onto the network.

Step 3 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.

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.

Prerequisites:

- 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.

Step 1 Open the ACE App wire sheet.

Step 2 Open the **ACE Catalog** sidebar.

Step 3 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.

Step 4 From the Ace Nrio subfolder in the sidebar, drop an NrioXXDevice onto the ACE App wiresheet, where NrioXXDevice is either Nrio16Device or Nrio34Device.

Step 5 Right-click the NrioDevice and click **Actions→Match**.

Step 6 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.

Adding Nrio Points

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

Prerequisites:

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.

Step 1 Open a **Property Sheet** (or **Wire Sheet**) view of the NrioDevice.

Step 2 From the **ACE Catalog**, drag a point of the desired type onto the view.

Step 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 ao1, you enter "1").
- For**Inputs**, select the input type (Resistance or Voltage).

Chapter 3 Components and views

Topics covered in this chapter

- aceEdge-AceEdgeNetwork
- ace-AceDevice
- ace-AcePointDeviceExt
- ace-Ace App
- ace-AceFolder
- ace-AcePointFolder
- ace-AceDynamicComp
- ace-AceCompManager
- ace-AcePointManager
- ace-AceWireSheet

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.

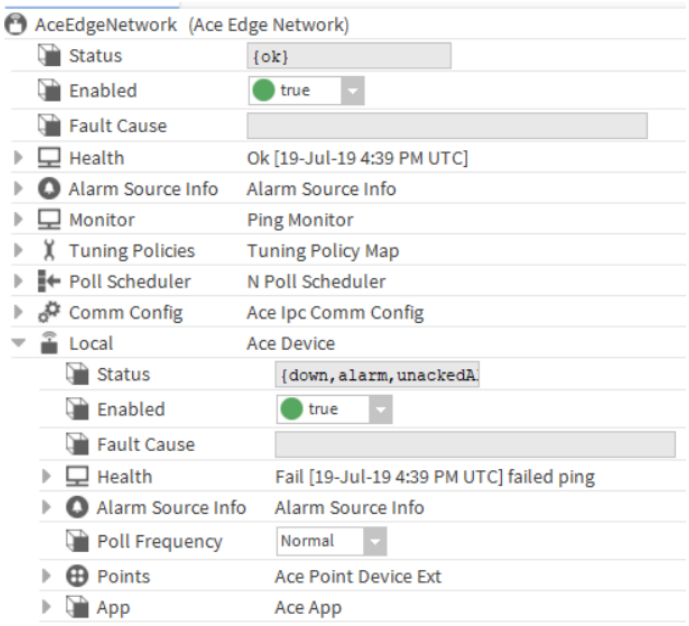
The component and view topics that follow appear as context-sensitive help topics when accessed by:

- 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.

Figure 9 AceEdgeNetwork Property Sheet view



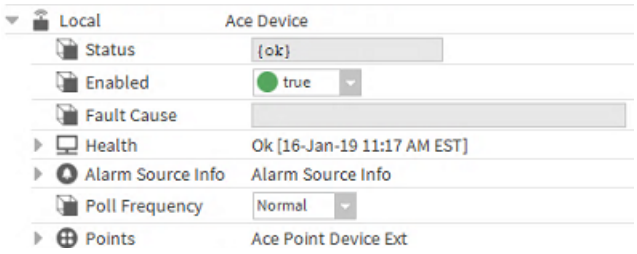
In addition to the standard network properties (Status, Enabled, Fault Cause, Health, etc.) the following configurable properties are present.

Type	Value	Description
Comm Config		Contains the Fault Cause subproperty

ace-AceDevice

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

Figure 10 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).

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.

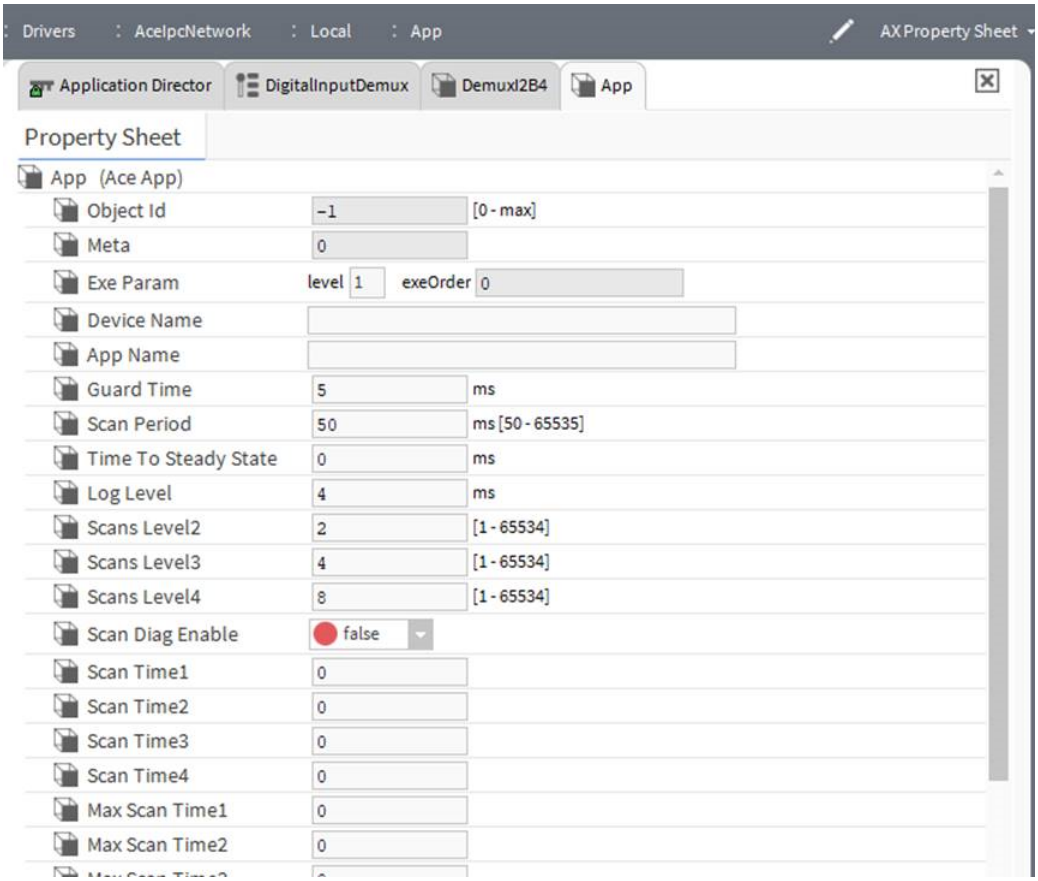
The Property Sheet view of the AcePointDeviceExt includes the following configurable Ace Point Discovery Preferences.

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

ace-Ace App

The ACE App runs in memory.

Figure 11 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		

Name	Value	Description
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.

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.

ace-AcePointFolder

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

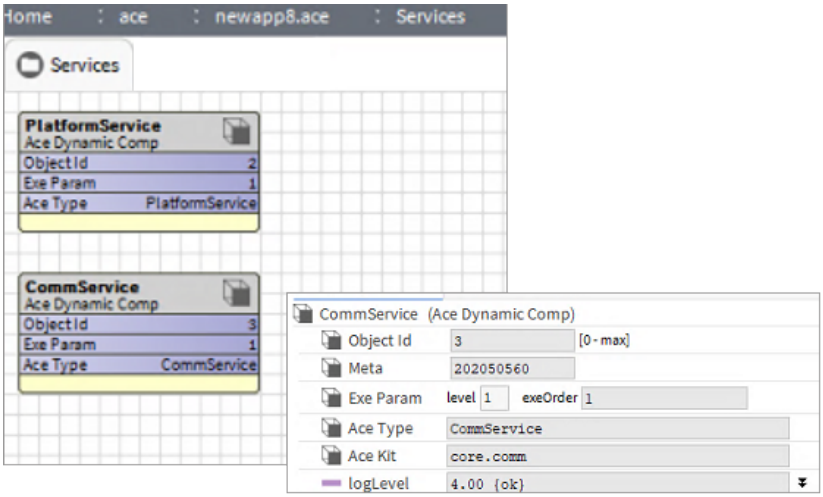
ace-AceDynamicComp

The AceDynamicComp, a hidden component, is a representation of a runtime ACE component containing a dynamically created list of 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 12 Common properties on AceDynamicComp objects



Common properties

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.
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 13 Example Select object link error

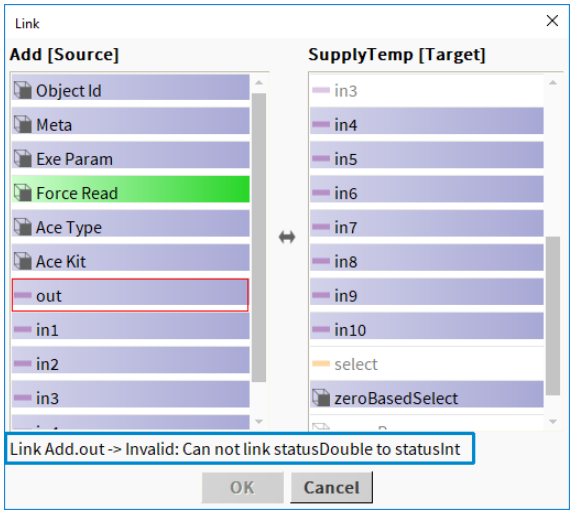
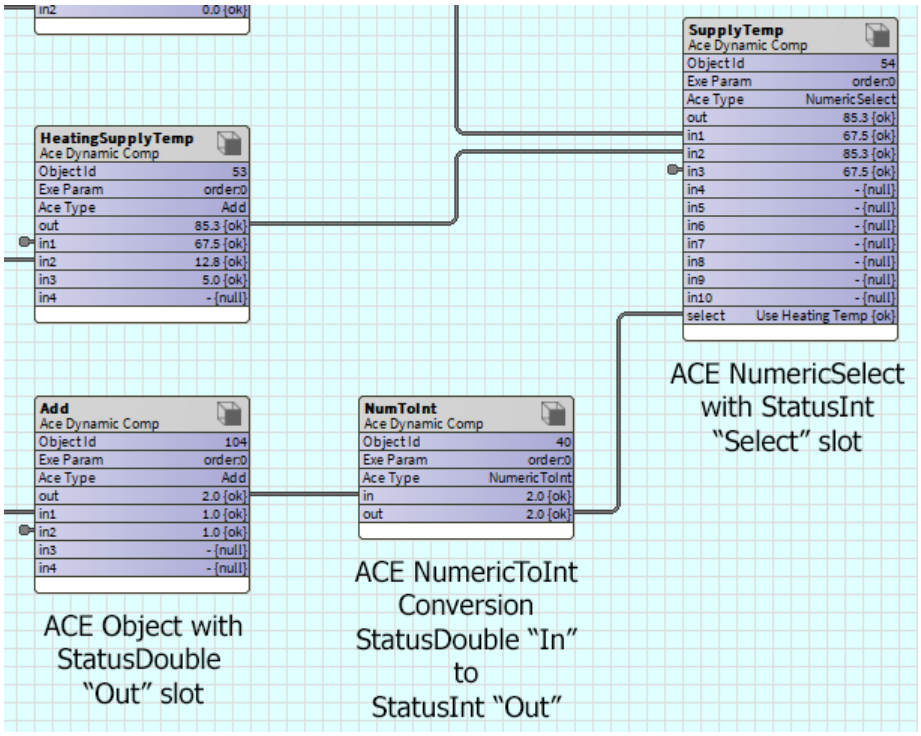
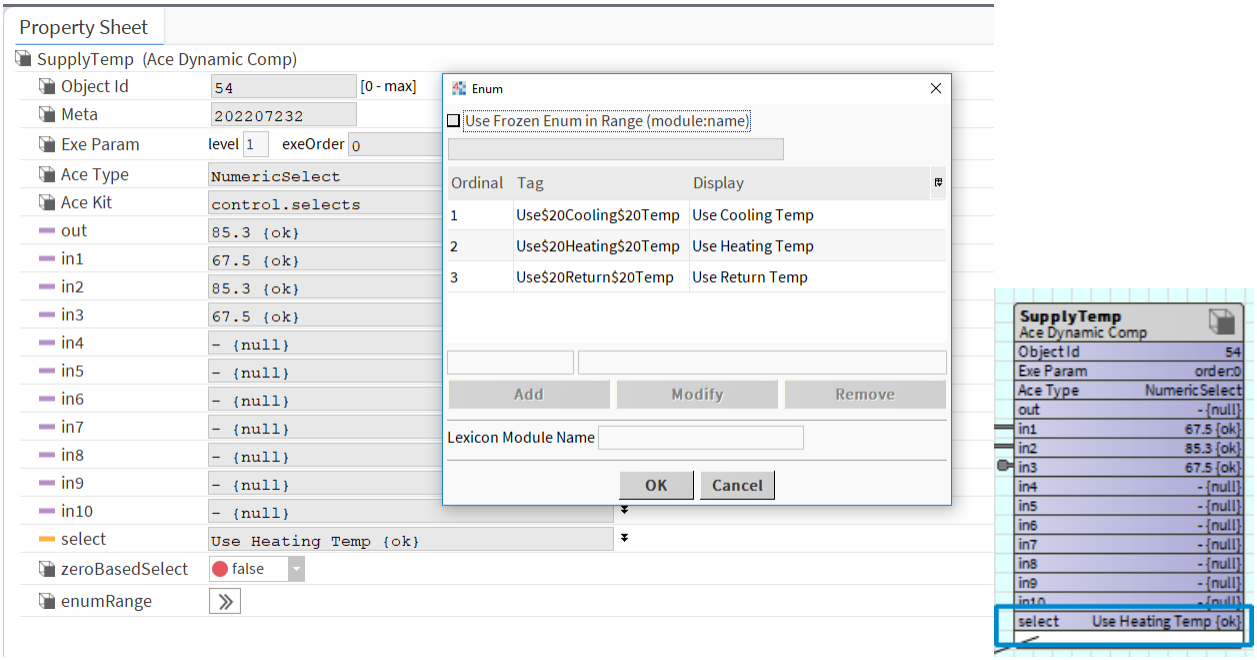


Figure 14 Example usage of NumericToInt conversion component



Also, for Select components there is a new enumRange slot at the bottom of the select objects which allows you to define what each select value represents.

Figure 15 enumRange slot at the bottom of the select object

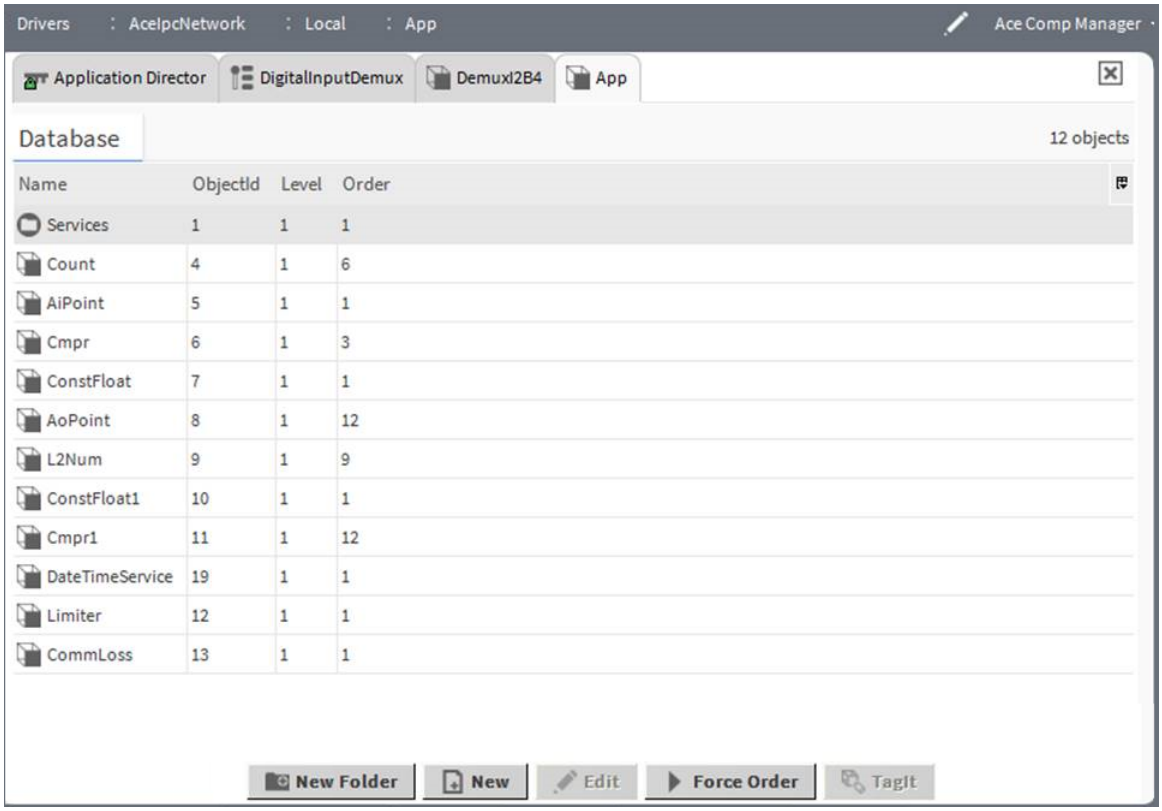


In the above example, when selecting between three inputs with select statusEnum values being `1`, `2` and `3`. The enumRange at the bottom is defined as `1=Off`, `2=Heating`, and `3=Return`. The select slot at the bottom of the object then shows which input is currently selected as Off, Cooling or Heating. Shown here, the numeric select is currently using the Heating input, or `in2`.

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.

Figure 16 Ace Comp Manager view shows components in the Ace App



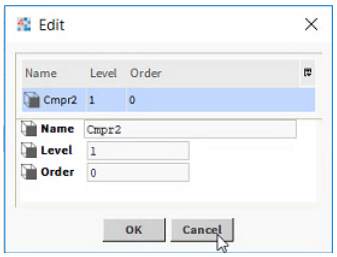
When the ACE App is running it constantly scans the components. The scan frequency is configurable via Level and Order settings. Scan "Level" is how you configure the frequency of component scanning . See Scan Period property (50 ms default) in the ACE App **Property Sheet** view. Scan Level 1 (not visible) 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.



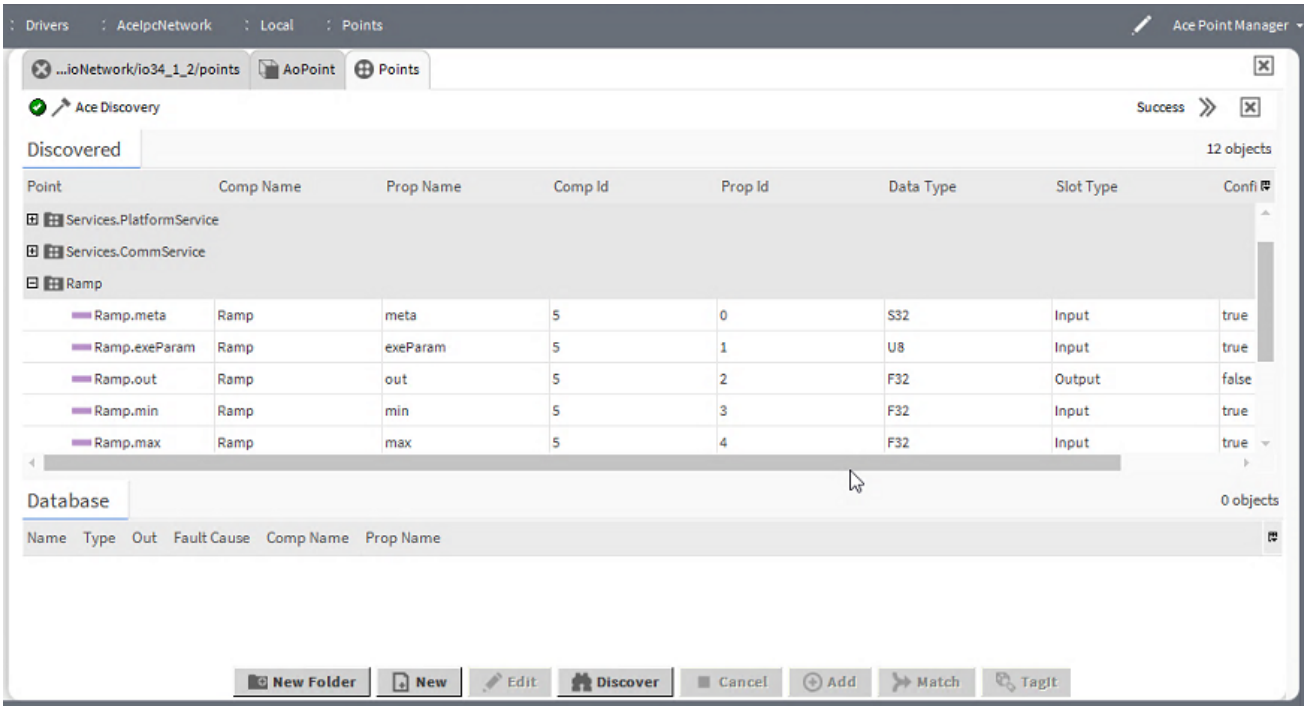
- Force Order — evaluates the components of the App and determines an execution order based on that.
- TagIt —

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

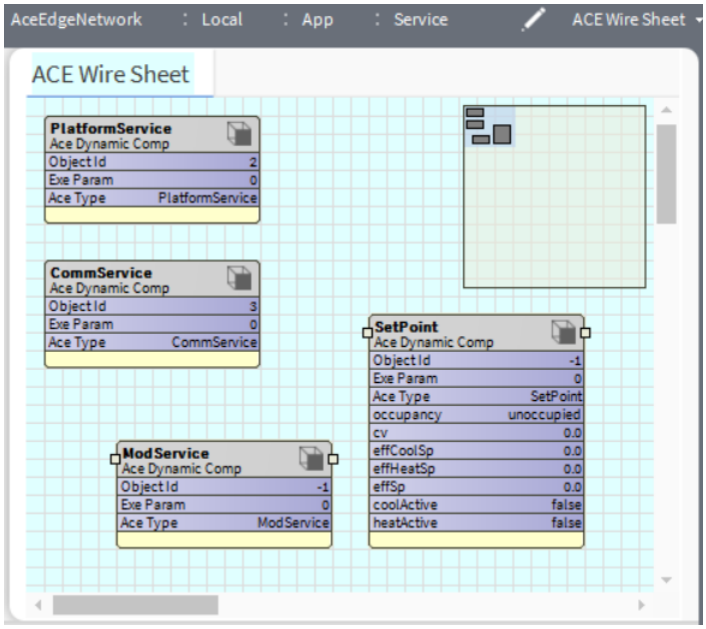
Figure 17 Ace Point Manager view showing expanded Ramp component properties



ace-AceWireSheet

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

Figure 18 ACE Wire Sheet view of the Service folder in the AceApp



Ace Dynamic Comp Index

Data Structure Index

a | b | c | d | e | f | g | h | i | l | m | n | o | p | r | s | t | u | w | x

a

AbsValue (control)
Add (control)
AnalogInput (tridium)
AnalogOutput (tridium)
And (control)
Average (control)

b

BooleanConst (control)
BooleanDelay (control)
BooleanDemux (control)
BooleanSelect (control)
BooleanSwitch (control)
BooleanWritable (control)

c

CommLoss (core)
CommService (core)
Conversion (tridium)
Counter (control)
CurrentTime (control)

d

DigitalInput (tridium)

g

GenericTblConv (tridium)
GreaterThan (control)
GreaterThanEqual (control)

h

Hysteresis (control)

i

IntegerSelect (control)
IntegerSwitch (control)

l

LessThan (control)
LessThanEqual (control)
Limiter (control)
Linearize (control)
LoopPoint (control)

m

Maximum (control)
Minimum (control)
MinMaxOverTime (control)
ModService (core)
Modulus (control)
Multiply (control)

MultiVibrator (control)

p

PlatformService (core)
Point (tridium)
Power (control)
PowerOnDelay (control)
Psychrometric (control)
PulseInputCount (tridium)
PulseInputFixedWindow (tridium)
PulseInputSlidingWindow (tridium)
PulseInputTrigger (tridium)

r

RaiseLower (control)
Ramp (control)
ReheatSequence (control)
Round (control)

s

SequenceLinear (control)
SetPoint (control)
SetResetLatch (control)
SineWave (control)
SquareRoot (control)
Subtract (control)

t

Therm10K3A1Conv (tridium)
Therm10K4A1Conv (tridium)
ThermType3Conv (tridium)

Ace Dynamic Comp Index June 2020

Niagara-4.9

Overview

The Autonomous Control Engine (ACE) is a standalone executable that runs on the same platform as a Niagara instance. It is targeted to applications that need faster start up and more real time control.

This document contains the detailed information about ACE.

This enabling software technology is designed to be ported to third party hardware edge devices like below.

- HVAC controller/devices (Actuator, VAV, FCU, etc.)
- Lighting controller/devices (Dimmer, Relays, etc)
- Variable Frequency Drives
- Solar Application (Inverter, Controller, etc.)
- Other IoT edge devices

Ace Dynamic Comp Index June 2020

Niagara-4.9

Related Pages

Here is a list of all related documentation pages:

Ace Dynamic Comp Index June 2020

Niagara-4.9

Data Structures

Here are the data structures with brief descriptions:

[detail level 1 2]

- N control
  - C AbsValue
  - C Add
  - C And
  - C Average
  - C BooleanConst
  - C BooleanDelay
  - C BooleanDemux
  - C BooleanSelect
  - C BooleanSwitch
  - C BooleanWritable
  - C Counter
  - C CurrentTime
  - C Divide
  - C EnumConst
  - C EnumToBoolean
  - C EnumWritable
  - C Equal
  - C Exponential
  - C Frequency
  - C GreaterThan
  - C GreaterThanEqual
  - C Hysteresis
  - C IntegerSelect
  - C IntegerSwitch
  - C LessThan
  - C LessThanEqual

- Limiter
- Linearize
- LoopPoint
- Maximum
- Minimum
- MinMaxOverTime
- Modulus
- Multiply
- MultiVibrator
- Negative
- Not
- NotEqual
- NumericConst
- NumericDelay
- NumericDemux
- NumericSelect
- NumericSwitch
- NumericToInt
- NumericWritable
- OneShot
- Or
- Power
- PowerOnDelay
- Psychrometric
- RaiseLower
- Ramp
- ReheatSequence
- Round
- SequenceLinear
- SetPoint
- SetResetLatch
- SineWave
- SquareRoot
- Subtract
- TimeAverage
- Timer
- ToggleLatch

- C Tstat
- C WritablePoint
- C Xor
- ▼ N core
  - C CommLoss
  - C CommService
  - C Folder
  - C ModService
  - C PlatformService
- ▼ N tridium
  - C AnalogInput
  - C AnalogOutput
  - C Conversion
  - C DigitalInput
  - C DigitalOutput
  - C GenericTblConv
  - C Point
  - C PulseInputCount
  - C PulseInputFixedWindow
  - C PulseInputSlidingWindow
  - C PulseInputTrigger
  - C Therm10K3A1Conv
  - C Therm10K4A1Conv
  - C ThermType3Conv
  - C UnitConv

Ace Dynamic Comp Index June 2020

Niagara-4.9

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- doEmergencySet(): **control::EnumWritable**, **control::NumericWritable**
- doManualSet(): **control::EnumWritable**, **control::NumericWritable**
- doSet(): **control::EnumConst**, **control::EnumWritable**, **control::NumericConst**, **control::NumericWritable**

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::AbsValue Class Reference

Detailed Description

AbsValue get the abs value of the input dout is the output of double data type fout is the output of float data type

Properties

status_double dout

Facets: `@readonly @summary`

status_float fout

Facets: `@readonly @summary`

status_double in

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::GenericTblConv Class Reference

Detailed Description

GenericTblConv performs linear conversion base on x and y entries. The x entries must be increasing. The tableSize will be set based on the number of increasing x entries.

Properties

double input[20]

Facets: `@config[1:20]`

double output[20]

Facets: `@config[1:20]`

int tableSize

Facets: `@readonly @summary`

buf faultCause[64]

Facets: `@readonly @string`

Ace Dynamic Comp Index June 2020

Niagara-4.9

core::PlatformService Class Reference

Detailed Description

Properties

buf platformId[32]

Facets: `@readonly @string`

buf platformVersion[16]

Facets: `@readonly @string`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Add Class Reference

Detailed Description

Add adds four double inputs and writes the results to a double output

out = in1 + in2 + in3 + in4

Properties

status_double out

Facets: `@readonly @summary`

status_double in[4]

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::GreaterThan Class Reference

Detailed Description

GreaterThan performs the operation A > B with a boolean result.

Properties

status_bool out

Facets: `@readonly @summary`

status_double inA

Facets: `@input @summary`

status_double inB

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::Point Class Reference

Detailed Description

Properties

int status

Default: `0`

Facets: `@fieldEditor = "ace:AceStatusFE" @readonly @summary`

int pointType

Default: `0`

Facets:

```
@config @enumList = "
{universal=0,universalOut=1,universalInput=2,analogOut=3,analogInput=4,digitalOut=5,digitalInput=6}"
@noProxy
```

int pointIndex

Default: `0`

Facets: `@config @max = 64 @min = 1 @noProxy @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::AnalogInput Class Reference

Public Member Functions

Actions

void	**doOverride** ()
void	**doAuto** ()

Detailed Description

Models an analog input point in a Device. Tolerance is the minimum change required to change the output. The value read from hardware must change by this amount before out is updated.

Properties

int pointType

Default: `2`
Facets: `@config @enumList = "{universalInput=2}" @noProxy @override`

status_double out

Facets: `@precision = 3 @readonly @summary`

int mode

Default: `0`
Facets: `@config @enumList = "{voltageIn=0,resistanceIn=2}"`

double tolerance

Default: `0.0`
Facets: `@config @min = 0 @precision = 3`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::GreaterThanEqual Class Reference

Detailed Description

GreaterThanEqual performs the operation A >= B with a boolean result.

Properties

status_bool out

Facets: `@readonly @summary`

status_double inA

Facets: `@input @summary`

status_double inB

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Power Class Reference

Detailed Description

Power object that do math function power

out = in1^in2

Properties

status_double out

Facets: `@readonly @summary`

status_double in1

Facets: `@input @summary`

status_double in2

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::AnalogOutput Class Reference

Public Member Functions

Actions

void	**doOverride** ()
void	**doAuto** ()

Detailed Description

Models an analog output point in a Device.

Properties

int pointType

Default: `3`
Facets: `@config @enumList = "{analogOut=3}" @noProxy @override`

double delta

Default: `0.01`
Facets: `@config`

double in

Facets: `@input @summary`

status_double out

Facets: `@readonly @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::PowerOnDelay Class Reference

Detailed Description

PowerOnDelay: indicate the time delay on start of ACE

`out = false when power up for less than the startDelay secondsout = true when power up for more than the startDelay seconds`

Properties

bool out

Facets: `@readonly @summary`

int startDelay

Default: `60`
Facets: `@config @summary @unit = "second"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::And Class Reference

Detailed Description

And 4 input AND logic object Null inputs are ignored. If all inputs are null output is null out = inA && inB && inC && inD

Properties

status_bool out

Facets: `@readonly @summary`

status_bool inA

Facets: `@input @summary`

status_bool inB

Facets: `@input @summary`

status_bool inC

Facets: `@input @summary`

status_bool inD

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Psychrometric Class Reference

Detailed Description

Psychrometric will provide the following output according to temprature and humidity: dew point temperature Enthalpy saturated pressure vapor pressure wet bulb temperature temperature unit is Fahrenheit by default. if isEnglish is false, then temperature unit is celcius.

Properties

bool isEnglish

Default: `true`
Facets: `@config`

status_double inTemp

Facets: `@input @summary`

status_double inHumidity

Facets: `@input @summary`

status_double outDewPoint

Facets: `@readonly @summary`

status_double outEnthalpy

Facets: `@readonly @summary`

status_double outSatPressure

Facets: `@readonly @summary`

status_double outVaporPressure

Facets: `@readonly @summary`

status_double outWetBulbTemp

Facets: `@readonly @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Average Class Reference

Detailed Description

Average determines the average value of valid inputs and writes that value to out. out = (in1 + in2 + in3 + in4) / 4

Properties

status_double out

Facets: `@readonly @summary`

status_double in[4]

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Hysteresis Class Reference

Detailed Description

Applies hysteresis to an input value to set a boolean out.

NOTE: If "Rising Edge" > "Falling Edge", then out behaves "normally", ie

```
out switches to "true" when in rises above "Rising Edge", and switches backto "false" when
in falls below "Falling Edge"Falling    RisingEdge       Edge|----------|-------|
|--------|----------|
```

NOTE: If "Rising Edge" < "Falling Edge", then out behaves "inverted", ie

```
out switches to "false" when in rises above "Falling Edge", and switches backto "true" when
in falls below "Rising Edge"Rising     FallingEdge        Edge--------|----------||
||----------|-------
```

NOTE: If "Rising Edge" == "Falling Edge", this object become a simple

```
comparator, where if in > Rising Edge, then out = true
```

Properties

double in

Facets: `@input @summary`

bool out

Default: `false`
Facets: `@readonly @summary`

double risingEdge

Default: `50.0`

Facets: `@config @defaultOnClone`

double fallingEdge

Default: `50.0`
Facets: `@config @defaultOnClone`

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::PulseInputCount Class Reference

Public Member Functions

Actions

void	**doOverride** ()
void	**doAuto** ()
void	**doSet** ()
void	**doReset** ()

Detailed Description

Models an pulse input point in a Device. The type of calculation is count.

Properties

int pointType

Default: `2`

Facets: `@config @enumList = "{universalInput=2}" @noProxy @override`

int mode

Default: `4`

Facets: `@config @enumList = "{pulseIn=4}"`

status_double total

Facets: `@readonly @summary`

int calcType

Default: `0`

Facets: `@enumList = "{count=0}" @readonly`

status_double out

Facets: `@readonly @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::PulseInputFixedWindow Class Reference

Detailed Description

Models an pulse input point in a Device. The Calc type is FixedWindow.

Properties

int calcType

Default: `1`
Facets: `@enumList = "{fixedWindowRate=1}" @override @readonly`

status_double rate

Facets: `@readonly @summary`

double rateScale

Default: `1.0`
Facets: `@config @precision = 5 @summary`

double rateInterval

Default: `60.0`
Facets: `@config @min = 1 @summary @unit = "second"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::PulseInputSlidingWindow Class Reference

Detailed Description

Models an pulse input point in a Device. The Calc type is SlidingWindow.

Properties

int calcType

Default: `2`

Facets: `@enumList = "{slidingWindowRate=2}" @override @readonly`

int rateWindows

Default: `6`

Facets: `@config @min = 2 @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::IntegerSelect Class Reference

Detailed Description

IntegerSelect Integer Select object

Selects one of inputs to route to output based on selector value1

Properties

status_int out

Facets: `@readonly @summary`

status_int in[10]

status_double select

Facets: `@input @summary`

bool zeroBasedSelect

Default: `false`
Facets: `@config @noProxy`

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::PulseInputTrigger Class Reference

Public Member Functions

Actions

void **doRecalculateRate** ()

Detailed Description

Models an pulse input point in a Device. The Calc type is Trigger.

Properties

int calcType

Default: `3`

Facets: `@enumList = "{triggerRate=3}" @override @readonly`

double rateInterval

Default: `60.0`

Facets: `@config @hidden @min = 1 @override @summary @unit = "second"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::IntegerSwitch Class Reference

Detailed Description

IntegerSwitch Integer Switch object

out = inSwitch ? inTrue : inFalse

Properties

status_int out

Facets: `@readonly @summary`

status_int inTrue

Facets: `@input @summary`

status_int inFalse

Facets: `@input @summary`

status_bool inSwitch

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::BooleanConst Class Reference

Public Member Functions

Actions

void	**doActive** ()
void	**doInactive** ()
void	**doSetNull** ()
void	**doSet** ()

Detailed Description

BooleanConst: boolean constant

`out should never be a link destination.`

Properties

status_bool out

Facets: `@config @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::RaiseLower Class Reference

Public Member Functions

Actions

void **doReset** ()

Detailed Description

RaiseLower is a floating output component When the input value increase, the raise output is true, when the input value decrease, the lower output is true. The user can configure the property "driveTime", which is a coefficient that calculate the time that the raise output and lower output keep true. If midnightResetEnabled is true and **CurrentTime** component is added, it will reset at midnightResetTime everyday.

Properties

double out

Facets: `@readonly @summary`

double in

Facets: `@input @max = 100 @min = 0 @summary`

double lastValidIn

Facets: `@hidden @max = 100 @min = 0`

double virtualPosition

Facets: `@readonly`

bool raise

Facets: `@operator @readonly`

bool lower

Facets: `@operator @readonly`

int function

Default: `0`
Facets:
```
@enumList = "
{offState=0,staticState=1,lowerState=2,raiseState=3,resetRaiseState=4,resetLowerState=5}"
@operator @readonly
```

double deadband

Default: `0.5`
Facets: `@config @max = 5 @min = 0`

int driveTimeMs

Default: `60000`
Facets: `@config @min = 1000 @unit = "millisecond"`

bool midnightResetEnabled

Default: `true`
Facets: `@config`

int midnightResetTime

Default: `1435`
Facets: `@config @hidden @unit = "minute"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::BooleanDelay Class Reference

Detailed Description

BooleanDelay provides a way to delay the status change.

The **BooleanDelay** component provides a way to delay the status change of a boolean status "out"property value by configuring an associated "Delay" property.Delay properties are provided for on (true) and off (false) statuses andare labeled "On Delay" and "Off Delay", respectively.The delay applies to any transition (status change from on to off or off to on)at the component's status boolean input.Both delay times are configurable in terms of hours, minutes and seconds.

Properties

status_bool out

Facets: `@readonly @summary`

status_bool outNot

Facets: `@readonly @summary`

status_bool in

Facets: `@input @summary`

int onDelay

Facets: `@config @min = 0 @unit = "second"`

int offDelay

Facets: `@config @min = 0 @unit = "second"`

bool onDelayActive

Facets: `@readonly @summary`

bool offDelayActive

Facets: `@readonly @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::LessThan Class Reference

Detailed Description

LessThan performs the operation A < B with a boolean result.

Properties

status_bool out

Facets: `@readonly @summary`

status_double inA

Facets: `@input @summary`

status_double inB

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Ramp Class Reference

Detailed Description

Ramp

`Ramp` provides a StatusNumeric Out with a linear ramping output.

Properties

bool enabled

Default: `true`
Facets: `@config`

int period

Default: `30000`
Facets: `@config @min = 50 @unit = "millisecond"`

double amplitude

Default: `50.0`
Facets: `@config`

double offset

Default: `50.0`
Facets: `@config`

int updateInterval

Default: `1000`
Facets: `@config @min = 50 @unit = "millisecond"`

status_double out

Facets: `@readonly @summary`

int waveform

Default: `0`

Facets: `@config @enumList = "{triangle=0,sawtooth=1,revertsawtooth=2}"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::BooleanDemux Class Reference

Detailed Description

Demuxes an integer count input into 4 boolean outputs. By setting "startsAt" appropriately, N objects can demux to N*4 outputs (ie, you can create a 4x, 8x, 16x, etc demux) if in == startsAt + 0, out1 is true, else false if in == startsAt + 1, out2 is true, else false if in == startsAt + 2, out3 is true, else false if in == startsAt + 3, out4 is true, else false

Properties

status_double in

Facets: `@input @summary`

status_bool out[4]

int startsAt

Default: `1`
Facets: `@config`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::LessThanEqual Class Reference

Detailed Description

LessThanEqual performs the operation A <= B with a boolean result.

Properties

status_bool out

Facets: `@readonly @summary`

status_double inA

Facets: `@input @summary`

status_double inB

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::ReheatSequence Class Reference

Detailed Description

ReheatSequence will provide a linear sequence of up to 4 loads based on configurable thresholds

Sets an output true if the "in" value is greater than corresponding threshold, and returns the ouput to false if the "in" value is less than threshold minus the hysteresis value.

When "delay" is configured to be 0, "currentOn" is the same as "desiredOn". When "delay" is configured to be greater than 0, "currentOn" is the count of outputs that are true (0 to 4), "desiredOn" is the count of outputs that are true when delayActive is false. If "enable" is false, all outputs are set to false regardless of in value.

Properties

bool out[4]

double in

Facets: `@input @summary`

bool enable

Facets: `@config`

int delay

Default: `0`
Facets: `@config @min = 0 @unit = "second"`

byte desiredOn

Default: `0`
Facets: `@readonly @summary`

byte currentOn

Default: `0`

Facets: `@readonly @summary`

double hysteresis

Facets: `@config`

double threshold[4]

bool delayActive

Facets: `@readonly @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::BooleanSelect Class Reference

Detailed Description

BooleanSelect: Boolean Select object

Selects one of inputs to route to output based on selector value

Properties

status_bool out

Facets: `@readonly @summary`

status_bool in[10]

status_double select

Facets: `@input @summary`

bool zeroBasedSelect

Default: `false`
Facets: `@config @noProxy`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Limiter Class Reference

Detailed Description

Limits input between two configured values If (in < lowLmt), out = lowLmt else if (in > highLmt), out = highLmt else out = in

Properties

status_double out

Facets: `@readonly @summary`

status_double in

Facets: `@input @summary`

double lowLmt

Facets: `@config`

double highLmt

Facets: `@config`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Round Class Reference

Detailed Description

Rounds a float to nearest n places. Uses the tie-breaker rule that positive number are rounded up if a tie-break situation exists, and negative numbers are rounded down ("more negative") if a tie-break situation exists.

Example: 123.456 with n = 0 : rounds to 123.000

```
with n = 1 : rounds to 123.500with n = -1: rounds to 120.000
```

Example: -123.456 with n = 0 : rounds to -123.000

```
with n = 1 : rounds to -123.500with n = -1: rounds to -120.000
```

```
Positive numbers:1) multiply by 10^n2) add 0.5 and truncate (convert to integer)3) divide by 10^n
```

```
Negative numbers:1) multiply by 10^n2) subtract 0.5 and truncate (convert to integer)3) divide by 10^n
```

```
where n = number of decimal places to round
```

```
out = round(in)
```

Properties

status_double out

Facets: `@precision = 3 @readonly @summary`

status_double in

Facets: `@input @precision = 3 @summary`

int decimalPlaces

Default: `0`

Facets: `@config @max = 3 @min = -1`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::BooleanSwitch Class Reference

Detailed Description

BooleanSwitch: Boolean Switch object

if inSwitch is null output is null if (inSwitch) out = inTrue else out = inFalse

Properties

status_bool out

Facets: `@readonly @summary`

status_bool inTrue

Facets: `@input @summary`

status_bool inFalse

Facets: `@input @summary`

status_bool inSwitch

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Linearize Class Reference

Detailed Description

Converts a table of values into a curve using linear interpolation between the values. Individual slope/intercept constants are computed between the x's and y's using the formula y = mx + b, where m = ym - yn/xm - xn

If in is not in the range of x0 to x9, then output is set to "nan"

Note that slope may be positive or negative, and is indicated by comparison of x1 and x0 (positive if x1 > x0, negitive if x1 < x0)

Properties

status_double out

Facets: `@readonly @summary`

status_double in

Facets: `@input @summary`

double x[10]

double y[10]

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::BooleanWritable Class Reference

Public Member Functions

Actions

void	**doEmergencyActive** ()
void	**doEmergencyInactive** ()
void	**doEmergencyAuto** ()
void	**doActive** ()
void	**doInactive** ()
void	**doAuto** ()
void	**doSet** ()

Detailed Description

BooleanWritable

Properties

status_bool out

Facets: `@readonly @summary`

status_bool in[16]

Facets:

`@input[7] @input[2:5] @input[9:16] @readonly[1] @readonly[6] @readonly[8] @summary[1] @summary[8] @summary[10] @summary[16]`

status_bool fallback

Facets: `@config`

int minActiveTime

Default: `0`
Facets: `@config @unit = "second"`

int minInactiveTime

Default: `0`
Facets: `@config @unit = "second"`

bool setMinInactiveTimeOnStart

Default: `false`
Facets: `@config`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::LoopPoint Class Reference

Detailed Description

Control loop feedback component.

Properties

bool loopEnable

Default: `true`
Facets: `@config`

status_double setpoint

Facets: `@input @precision = 3 @summary`

status_double controlledVariable

Facets: `@input @precision = 3 @summary`

status_double out

Facets: `@readonly @summary`

double proportionalConstant

Default: `1.0`
Facets: `@config @min = 0 @precision = 6`

double integralConstant

Default: `0.0`
Facets: `@config @min = 0 @precision = 6 @unit = "per minute"`

double derivativeConstant

Default: `0.0`
Facets: `@config @min = 0 @precision = 6 @unit = "second"`

double max

Default: `100.0`
Facets: `@config @precision = 6`

double min

Default: `0.0`
Facets: `@config @precision = 6`

double bias

Default: `0.0`
Facets: `@config @precision = 6`

double maxDelta

Default: `0.0`
Facets: `@config @min = 0 @precision = 6`

bool direct

Default: `true`
Facets: `@config`

int exTime

Default: `1000`
Facets: `@config @min = 0 @unit = "millisecond"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::SequenceLinear Class Reference

Detailed Description

SequenceLinear will provide a linear sequence of 1 to 10 loads.

Analogous to a bar graph of the input value, where the delta represented by each output is (inMax-inMin)/(numOuts). So given an input value, outputs 1 through n will be set true, and any remaining outputs will be false.

If 'in' > 'inMax', then 'numOuts' outputs will be set true, and 'ovfl' will be set true The range of 'inMin' to 'inMax' is divided into 'numOuts' threshold values (delta), where:

```
'out1' is driven true if in > inMin + 1*delta,'out1' and 'out2' are driven true if in > inMin + 2*delta,'out1','out2', and 'out3' are driven true if in > inMin + 3*delta,etcdelta = inMax-inMin/(numOuts)
```

A hysteresis of 1/2 delta is required to turn an output off. If delay > 0, it will delay configured seconds to turn on or turn off an output. If isRotating is true, outputs will go around from output 1 to numOuts, then go back to 1.

Properties

double in

Facets: `@input @summary`

double inMin

Default: `0.0`
Facets: `@config`

double inMax

Default: `100.0`
Facets: `@config`

int numOuts

Default: `10`
Facets: `@config @max = 10 @min = 2 @noProxy`

int delay

Default: `0`
Facets: `@config @min = 0 @unit = "second"`

double delta

Facets: `@readonly @summary`

byte desiredOn

Facets: `@readonly @summary`

byte currentOn

Facets: `@readonly @summary`

bool out[10]

Facets: `@summary[1:3]`

bool ovfl

Facets: `@readonly @summary`

bool onDelayActive

Facets: `@readonly @summary`

bool offDelayActive

Facets: `@readonly @summary`

bool isRotating

Default: `false`
Facets: `@config`

int rotateTime

Default: `1`
Facets: `@config @min = 0 @unit = "second"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::SetPoint Class Reference

Detailed Description

SetPoint calculate effective setpoint based on occupancy and control variable

Properties

double unocCoolSp

Default: `30.0`
Facets: `@config`

double stbyCoolSp

Default: `30.0`
Facets: `@config`

double occCoolSp

Default: `30.0`
Facets: `@config`

double unocHeatSp

Default: `0.0`
Facets: `@config`

double stbyHeatSp

Default: `0.0`
Facets: `@config`

double occHeatSp

Default: `0.0`

Facets: `@config`

double minDiff

Default: `2.0`
Facets: `@config`

double spOffset

Default: `0.0`
Facets: `@config`

int occupancy

Default: `0`
Facets: `@enumList = "{unoccupied=0,occupied=1,standby=2}" @input @summary`

double cv

Default: `0.0`
Facets: `@input @summary`

double effCoolSp

Facets: `@readonly @summary`

double effHeatSp

Facets: `@readonly @summary`

double effSp

Facets: `@readonly @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

core::CommLoss Class Reference

Detailed Description

CommLoss will provide a logical indication of loss of comm connections. A commLossDelay property allows delay till indication of commLoss.

Properties

bool commLoss

Facets: `@readonly @summary`

int commLossDelay

Facets: `@config @min = 0 @unit = "second"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Maximum Class Reference

Detailed Description

Maximum selects the highest of 2 inputs to send to the output

Properties

status_double out

Facets: `@readonly @summary`

status_double in[4]

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::SetResetLatch Class Reference

Detailed Description

SetResetLatch Set Reset Latch object

if ^set out = true; if ^reset out = false; If both inputs have rising edges, out becomes false

Properties

bool out

Facets: `@readonly @summary`

bool set

Facets: `@input @summary`

bool reset

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

core::CommService Class Reference

Detailed Description

CommService is a service for ACE communication.

Properties

byte logLevel

Default: `4`

Facets: `@config @enumList = "|ace:AceLogLevelEnum"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Minimum Class Reference

Detailed Description

Minimum selects the lowest of 4 inputs to send to the output

Properties

status_double out

Facets: `@readonly @summary`

status_double in[4]

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::SineWave Class Reference

Detailed Description

SineWave generates a sine wave as a StatusNumeric out.

Properties

bool enabled

Default: `true`
Facets: `@config`

int period

Default: `30000`
Facets: `@config @min = 50 @unit = "millisecond"`

double amplitude

Default: `50.0`
Facets: `@config`

double offset

Default: `50.0`
Facets: `@config`

int updateInterval

Default: `1000`
Facets: `@config @min = 50 @unit = "millisecond"`

status_double out

Facets: `@readonly @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::Conversion Class Reference

Detailed Description

Properties

double in

Facets: `@input @precision = 3 @summary`

double outC

Facets: `@readonly @summary @unit = "celsius"`

double outF

Facets: `@readonly @summary @unit = "fahrenheit"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::MinMaxOverTime Class Reference

Public Member Functions

Actions

void **doReset** ()

Detailed Description

MinMaxOverTime object computes min and max of an input value every execute cycle Resets **MinMaxOverTime** to input value if r == true

Properties

status_double minOut

Facets: `@readonly @summary`

status_double maxOut

Facets: `@readonly @summary`

status_double in

Facets: `@input @summary`

bool r

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::SquareRoot Class Reference

Detailed Description

Calculate square root of the input

out = square root of in1

Properties

status_double out

Facets: `@readonly @summary`

status_double in

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Counter Class Reference

Public Member Functions

Actions

void **doPreset** ()

Detailed Description

pulse counter object, counts transitions form 0 to 1 of input "in" Counts up if dir == true, counts down if dir == false Forced to preset value if enable = false In the case where dir == false, the counter will stop counting down at 0

Properties

double out

Facets: `@readonly @summary`

bool in

Facets: `@input @summary`

int presetValue

Default: `0`
Facets: `@config @defaultOnClone @min = 0`

bool dir

Default: `true`
Facets: `@config @defaultOnClone @falseText = "down" @trueText = "up"`

bool enable

Default: `true`

Facets: `@input @summary`

bool presetIn

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

core::ModService Class Reference

Detailed Description

Properties

bool enable

Default: `true`
Facets: `@config`

int maxUpdateTime

Default: `5`
Facets: `@config @unit = "second"`

byte logLevel

Default: `1`
Facets: `@config @enumList = "|ace:AceLogLevelEnum"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Subtract Class Reference

Detailed Description

Subtract two Input **Subtract** object

out = in1 - in2

Properties

status_double out

Facets: `@readonly @summary`

status_double in1

Facets: `@input @summary`

status_double in2

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::CurrentTime Class Reference

Public Member Functions

Actions

void **doRefresh** ()

Detailed Description

CurrentTime reads system time and translate it to local time. It is automatically updated each second.

Properties

long nanos

Facets: `@readonly @unit = "nanosecond"`

byte hour

Facets: `@readonly`

byte minute

Facets: `@readonly`

byte second

Facets: `@readonly`

int year

Facets: `@readonly`

byte month

Facets: `@enumList = "|baja:Month" @readonly`

byte day

Facets: `@readonly`

byte dayOfWeek

Facets: `@enumList = "|baja:Weekday" @readonly`

bool useUtcOffset

Facets: `@config`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Modulus Class Reference

Detailed Description

Modulus class calculates the remainder when we divide input1 by input2.

Properties

status_double out

Facets: `@readonly @summary`

status_double in1

Facets: `@input @summary`

status_double in2

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Multiply Class Reference

Detailed Description

Multiply Four Input Multiplication object

out = in1 * in2 * in3 * in4

Properties

status_double out

Facets: `@readonly @summary`

status_double in[4]

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::MultiVibrator Class Reference

Detailed Description

MultiVibrator component

```
Generates a repeating pulse train with period and high dutyCycle
```

Properties

bool out

Facets: `@readonly @summary`

int period

Default: `1000`
Facets: `@config @max = 86400000 @min = 200 @unit = "millisecond"`

int dutyCycle

Default: `50`
Facets: `@config @max = 100 @min = 0`

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::Therm10K3A1Conv Class Reference

Detailed Description

Properties

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::Therm10K4A1Conv Class Reference

Detailed Description

Properties

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::DigitalInput Class Reference

Public Member Functions

Actions

void	**doOverride** ()
void	**doAuto** ()

Detailed Description

Models an digital input point in a Device.

Properties

int pointType

Default: `2`
Facets: `@config @enumList = "{universalInput=2}" @noProxy @override`

bool polarity

Default: `false`
Facets: `@config @falseText = "normal" @trueText = "reverse"`

status_bool out

Facets: `@readonly @summary`

int mode

Default: `0`
Facets: `@config @enumList = "{voltageIn=0,resistanceIn=2}"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::ThermType3Conv Class Reference

Detailed Description

Properties

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::DigitalOutput Class Reference

Public Member Functions

Actions

void	**doOverride** ()
void	**doAuto** ()

Detailed Description

Models an digital output point in a Device.

Properties

int pointType

Default: `5`

Facets: `@config @enumList = "{digitalOut=5}" @noProxy @override`

bool polarity

Default: `false`

Facets: `@config @falseText = "normal" @trueText = "reverse"`

int minOn

Default: `0`

Facets: `@config @min = 0 @unit = "millisecond"`

int minOff

Default: `0`

Facets: `@config @min = 0 @unit = "millisecond"`

bool in

Facets: `@input @summary`

status_bool out

Facets: `@readonly @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::TimeAverage Class Reference

Public Member Functions

Actions

void **doReset** ()

Detailed Description

TimeAverage object averages "in" over the configured time.

Note that this is NOT a running or moving average - this object caches the average over the previous time as the out value, and updates out every "time" ms.

Until the first full cycle has elapsed, out is set to average of all samples so far.

The average may be reset/restarted at any time using the "reset" action.

Properties

status_double out

Facets: `@readonly @summary`

status_double in

Facets: `@input @summary`

int time

Default: `10000`
Facets: `@config @summary @unit = "millisecond"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Divide Class Reference

Detailed Description

Divide Two Input Division object

out = in1 / in2

Properties

status_double out

Facets: `@readonly @summary`

status_double in1

Facets: `@input @summary`

status_double in2

Facets: `@input @summary`

bool div0

Facets: `@readonly @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Negative Class Reference

Detailed Description

Negative Negate math object

out = -in

Properties

status_double out

Facets: `@readonly @summary`

status_double in

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Timer Class Reference

Public Member Functions

Actions	
void	**doResetTimer** ()
void	**doStartTimer** ()

Detailed Description

Timer outputs a pulse for the configured amount of time "run" is used to fire the timer:

```
- if false, out is forced to false- if true, out = 1 until timer reaches "time"
secondsAlternatively, the pulse can be fired from the "Start Timer" action if run is not
linked.
```

Properties

bool out

Facets: `@readonly @summary`

bool run

Facets: `@falseText = "stop" @input @summary @trueText = "run"`

int time

Facets: `@config @summary @unit = "second"`

int left

Facets: `@readonly @summary @unit = "second"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Not Class Reference

Detailed Description

Not logic object

if in == null, out = null else out = !in

Properties

status_bool out

Facets: `@readonly @summary`

status_bool in

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::ToggleLatch Class Reference

Detailed Description

ToggleLatch component

`ToggleLatch shall toggle its "out" on each false to true transition of "in".An "enable" input will freeze/unfreeze transitions.`

Properties

bool out

Facets: `@readonly @summary`

bool in

Facets: `@input @summary`

bool enable

Default: `true`
Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::NotEqual Class Reference

Detailed Description

NotEqual performs the operation A != B with a boolean result.

Properties

status_bool out

Facets: `@readonly @summary`

status_double inA

Facets: `@input @summary`

status_double inB

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Tstat Class Reference

Detailed Description

Tstat Provides On/Off or Floating Thermostat control

```
raise & lower outputs operate independently of isHeating mode.lower will be set to true if cv > (sp + diff/2) and willremain set until cv < (sp - diff/2).raise will be set to true if cv < (sp - diff/2) and willremain set until CV > (sp + diff/2).

out will be set based on isHeating mode.if isHeatingout = raiseelseout = lower
```

Properties

double diff

Default: `0.0`
Facets: `@config @summary`

bool isHeating

Default: `false`
Facets: `@config @summary`

double sp

Facets: `@config @summary`

status_double cv

Facets: `@input @summary`

status_bool out

Facets: `@readonly @summary`

status_bool raise

Facets: `@readonly @summary`

status_bool lower

Facets: `@readonly @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::EnumConst Class Reference

Public Member Functions

Actions

void **doSet** ()

void **doSetNull** ()

Detailed Description

EnumConst: enum constant

`out should never be a link destination.`

Properties

status_int out

Facets: `@config @enumList = "::enumRange" @summary`

buf enumRange[128]

Facets: `@config @noProxy @string`

Member Function Documentation

doSet()

void doSet ( )

Facets: `@defaultParam = "out"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::NumericConst Class Reference

Public Member Functions

Actions

void	**doSet** ()
void	**doSetNull** ()

Detailed Description

NumericConst: numeric constant

```
out should never be a link destination.
```

Properties

status_double out

Facets: `@config @summary`

Member Function Documentation

doSet()

void doSet ( )

Facets: `@defaultParam = "out"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::EnumToBoolean Class Reference

Detailed Description

EnumToBoolean convert StatusEnum to StatusBoolean activeValues should be like "1,4,9" to make enum list. only decimal number is supported. activeValues does not support some input like "0x1", "d1" and "1f".

Properties

status_int in

Facets: `@input @summary`

status_bool out

Facets: `@readonly @summary`

buf activeValues[128]

Facets: `@config`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::NumericDelay Class Reference

Detailed Description

The **NumericDelay** component provides a "soft ramp" delay from StatusNumeric In to Out.

The **NumericDelay** component provides a "soft ramp" delay from StatusNumeric In to Out.The component uses configurable values in properties Max Step Size and Update Time toprovide a "stepped" output value.The combination of these two property values determines how quickly andhow smoothly the current Out value changes as it approaches the In value.

Properties

status_double out

Facets: `@readonly @summary`

status_double in

Facets: `@input @summary`

int updateTime

Facets: `@config @min = 0 @unit = "second"`

double maxStepSize

Facets: `@config`

Ace Dynamic Comp Index June 2020

Niagara-4.9

tridium::UnitConv Class Reference

Detailed Description

Properties

double in

Facets: `@input @summary`

double out

Facets: `@readonly @summary`

double scale

Default: `1.0`
Facets: `@config`

double offset

Default: `0.0`
Facets: `@config`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::EnumWritable Class Reference

Public Member Functions

Actions

void	**doEmergencySet** ()
void	**doEmergencyAuto** ()
void	**doManualSet** ()
void	**doManualAuto** ()
void	**doSet** ()

Detailed Description

EnumWritable

Properties

buf enumRange[128]

Facets: `@config @noProxy @string`

status_int out

Facets: `@enumList = "::enumRange" @readonly @summary`

status_int in[16]

Facets:
`@input[7] @input[2:5] @input[9:16] @readonly[1] @readonly[6] @readonly[8] @summary[1] @summary[8] @summary[10] @summary[16] @enumList[1:16] = "::enumRange"`

status_int fallback

Facets: `@config @enumList = "::enumRange"`

Member Function Documentation

doEmergencySet()

void doEmergencySet ( )

Facets: `@defaultParam = "in1"`

doManualSet()

void doManualSet ( )

Facets: `@defaultParam = "in8"`

doSet()

void doSet ( )

Facets: `@defaultParam = "fallback"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::NumericDemux Class Reference

Detailed Description

NumericDemux object selects one of two outputs to receive the input value, depending on the value of boolean switch input. The value of the other output remains unchanged.

If s1 is false, then out1 is updated with in, and out2 value remains constant at previous value If s2 is true, then out2 is updated with in, and out1 value remains constant at previous value

Properties

status_double out1

Facets: `@readonly @summary`

status_double out2

Facets: `@readonly @summary`

status_double in

Facets: `@input @summary`

status_bool s1

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Equal Class Reference

Detailed Description

Equal performs the operation A == B.

Properties

status_bool out

Facets: `@readonly @summary`

status_double inA

Facets: `@input @summary`

status_double inB

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::NumericSelect Class Reference

Detailed Description

Selects one of inputs to route to output based on selector value

Properties

status_double out

Facets: `@readonly @summary`

status_double in[10]

status_double select

Facets: `@input @summary`

bool zeroBasedSelect

Default: `false`
Facets: `@config @noProxy`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::WritablePoint Class Reference

Detailed Description

WritablePoint

Properties

int sourceLevel

Default: `17`

Facets: `@readonly`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Exponential Class Reference

Detailed Description

Calculates the base-e exponential function of the input. If the magnitude of the result is too large to be represented by a value of the return type, the output will be HUGE_VAL (or HUGE_VALF or HUGE_VALL) with the proper sign.

Properties

status_double out

Facets: `@readonly @summary`

status_double in1

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::NumericSwitch Class Reference

Detailed Description

NumericSwitch Switch object switches one of two inputs to the output

out = inSwitch ? inTrue : inFalse

Properties

status_double out

Facets: `@readonly @summary`

status_double inTrue

Facets: `@input @summary`

status_double inFalse

Facets: `@input @summary`

status_bool inSwitch

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::NumericToInt Class Reference

Detailed Description

NumericToInt converts a statusDouble input to a statusInt output. Input status is transferred to ouput. The input value is constrained to limits of a 32bit int.

Properties

status_double in

Facets: `@input @summary`

status_int out

Facets: `@readonly @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::NumericWritable Class Reference

Public Member Functions

Actions

void	**doEmergencySet** ()
void	**doEmergencyAuto** ()
void	**doManualSet** ()
void	**doManualAuto** ()
void	**doSet** ()

Detailed Description

NumericWritable

Properties

status_double out

Facets: `@readonly @summary`

status_double in[16]

Facets:

`@input[7] @input[2:5] @input[9:16] @readonly[1] @readonly[6] @readonly[8] @summary[1] @summary[8] @summary[10] @summary[16]`

status_double fallback

Facets: `@config`

Member Function Documentation

doEmergencySet()

void doEmergencySet ( )

Facets: `@defaultParam = "in1"`

doManualSet()

void doManualSet ( )

Facets: `@defaultParam = "in8"`

doSet()

void doSet ( )

Facets: `@defaultParam = "fallback"`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Xor Class Reference

Detailed Description

Xor Exclusive **Or** logic object

if either input null output is null out = inA xor inB

Properties

status_bool out

Facets: `@readonly @summary`

status_bool inA

Facets: `@input @summary`

status_bool inB

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

core::Folder Class Reference

Detailed Description

Properties

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Or Class Reference

Detailed Description

Or 4 input OR logic object null inputs ignored. If all inputs null, output is null out = inA || inB || inC || inD

Properties

status_bool out

Facets: `@readonly @summary`

status_bool inA

Facets: `@input @summary`

status_bool inB

Facets: `@input @summary`

status_bool inC

Facets: `@input @summary`

status_bool inD

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::Frequency Class Reference

Detailed Description

Calculate the frequency of the input pulse

Properties

status_double pps

Facets: `@precision = 3 @readonly @summary @unit = "per second"`

status_double ppm

Facets: `@precision = 3 @readonly @summary @unit = "per minute"`

status_bool in

Facets: `@input @summary`

Ace Dynamic Comp Index June 2020

Niagara-4.9

control::OneShot Class Reference

Public Member Functions

Actions

void **doClear** ()

Detailed Description

OneShot: Boolean one-shot pulse generator

```
out = true for pulseWidth sec, beginning at rising edge of inpulse retriggers on each
rising edge of in, if canRetrig = true
```

Properties

bool out

Facets: `@readonly @summary`

bool in

Facets: `@input @summary`

int pulseWidth

Default: `500`
Facets:
`@config @fieldEditor = "ace:AceRelTimeFE" @max = 86400000 @min = 1 @showMilliseconds @unit = "millisecond"`

bool canRetrig

Facets: `@config`

Ace Dynamic Comp Index June 2020

Niagara-4.9

AceEdge Niagara Driver

The aceEdge driver is a Niagara driver with two components: ace-rt.jar, ace-wb.jar, ace-ux.jar and aceEdge-rt.jar. It allows a station to connect the ACE instance for control monitoring.

The nem driver will currently allow the following operations:

- Download/Upload an application.
- Discovery of data points and addition of proxy points.
- Live program and applications

To use

The aceEdge driver follows the patten used in other Niagara Drivers. To setup a device do the following steps:

- From the palette sidebar select the aceEdge module.
- Add a AceEdgeNetwork to the Drivers

Application Creation

To create a new ACE application:

1. Open the Niagara Workbench Tools pull down
2. Select New ACE App.
3. Fill in the storage directory, the app a name > file to use.
4. From Workbench Window/Side Bars select ACE Catalog for palette of > components.
5. Drag components to wiresheet and link
6. Save to create a *.ace

Application Download

To download an application:

1. On the local ACE application / Download App
2. Navigate to the desired *.ace file and press ok
3. Observe success of failure reason in Jobs sidebar.

Add Proxy Points

Proxy points can be added to configure or monitor points in the device. Points are specific object/properties.

To add proxy points:

1. Navigate to the device's point folder.
2. Press the Discover button to populate discover pane with points. Points are organized in folders by component.
3. Select one or more points to add and press Add for a popup to select point name and point type. The default values are usually sufficient. You can also use hot key to add with default values – select one or more points and press 'a'.

Live Programming

1. On the local ACE application / View App
2. This will add and App component to Local and hyperlink to its wiresheet view.
3. Add components and links as desired.

Ace Dynamic Comp Index June 2020

Niagara-4.9

- doEmergencySet(): **control::EnumWritable**, **control::NumericWritable**
- doManualSet(): **control::EnumWritable**, **control::NumericWritable**
- doSet(): **control::EnumConst**, **control::EnumWritable**, **control::NumericConst**, **control::NumericWritable**

Index