Technical Document

Niagara Cloud Honeywell Sentience Driver Guide

December 12, 2022

niagara4

Niagara Cloud Honeywell Sentience Driver Guide

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

Confidentiality

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

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

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

Trademark notice

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

Copyright and patent notice

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

Copyright © 2023 Tridium, Inc. All rights reserved.

The product(s) described herein may be covered by one or more U.S. or foreign patents of Tridium.

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 how to use the Niagara Cloud (and CloudConnector component) to send data from a Niagara 4 JACE-8000 or Supervisor station into the cloud, facilitating data retrieval via web portals residing on a cloud platform. Included are descriptions of how to install and configure a NiagaraCloudNetwork.

Document change log

Updates (changes and additions) to this document are listed below.

December 12, 2022

- Added property descriptions for "Force Relinquish Enabled" and "Force Write Delay" in the "Cloud Point Device Ext (nCloudDriver-CloudPointDeviceExt)" component.

June 13, 2022

- Updated "Version compatibility matrix".
- Added property description for "Point Subscription Period".
- Updated "MessageClientUpgrader (cloudIotHubDep-MessageClientUpgrader)" component.
- Added chapter "From the cloud to station", and components "Cloud Write Controller" (nCloudDriver-CloudWriteController) and "Cloud Write Info (nCloudDriver-CloudWriteInfo).

April 20, 2021

- Edits to several component topics to update for Niagara 4.11/Honeywell Forge Connect FC 1.3.
- Updated Tuning Policies for the NiagaraCloudNetwork component, as well as several minor edits throughout.
- Updated the "Requirements" topic with a list of software modules required for any Forge Connect installation.
- Updated the "Event messages and system commands" topic with added details on new commands and changes to existing commands. Plus similar updates to "Predefined commands" and "Predefined command responses" topics in the "Security recommendations" section of the guide.

November 24, 2020

- Edits throughout to update for Niagara 4.9 /4.10 and Honeywell Forge Connect.
- Added "Troubleshooting" and "Tuning" chapters to this guide.
- In the "Install and configure" section, added a procedure on "Configuring a cloud Point Learn job". Other changes include, edits made in several of the component topics. Edited the section on, "Security recommendations".

August 12, 2019

- Edited an existing note in the "Requirements" topic to include the transparent proxy server requirement.

- Edited several component topics to add information on new properties for configuring message throttling.

December 17, 2018

- Added a note on trimming whitespace from JSON messages to the topic, "Send Commands".
- Added the procedure, "Configuring a history export learn job".
- Minor changes the topic, "cloudConnector-CloudConnector".
- Edited properties in the topics, "nCloudDriver-CloudHistoryExport", and "nCloudDriver-CloudProxyExt".
- Added the topic, "nCloudDriver-CloudHistoryExportManager.
- Edited steps and referenced palette in the procedure, "Adding the CloudConnector_Sentience service".
- Minor changes throughout the document related to the 2.0 release.
- Minor corrections throughout Also edited the topic, "User Authentication/Authorization".
- Edited the procedure, "Adding the CloudConnector_Sentience service".
- Many additional changes throughout related to added features for the 2018.5.x release. In the "Components and other references" chapter, added several new components topics Also added the chapter, "Security recommendations".
- Added two topics: "Configuring role mappings" and " nCloudDriver-RoleMappings". Also, a few changes were made to the cloudConnector-CloudConnector_Sentience nCloudDriver component topic.
- Many additional changes throughout related to added features for the 2018.4.x release.
- Minor changes to the "nCloudDriver-NiagaraCloudNetwork" and "nCloudDriver-CloudPointDeviceExt" component topics (added details on tuning policies).
- Minor changes to the "Requirements" topic, and added a procedure for "Installing software modules" to Chapter 1.
- Added the "Configuring the station to receive commands" topic and subtopics to Chapter 1.
- Added the "nCloudDriver-CloudAlarmExt" component topic to Chapter 3.
- Minor corrections to property descriptions for several component topics in Chapter 3.
- Many changes to component topics in Chapter 3.
- Minor updates, the "nCloudDriver-CloudAlarmRecipient" topic added to Chapter 3.
- Initial document publication

Related documentation

Additional information is available in the following documents.

- *Niagara 4 Hardening Guide* available from *https://www.tridium.com/us/en/services-support/library*
- *Niagara Asset Manager Guide* available from *https://www.tridium.com/us/en/services-support/library*
- *Niagara Drivers Guide*

Chapter 1 Getting started

Topics covered in this chapter

- Overview
- Requirements
- Installing software modules
- Setting up device internet access

This section provides an overview of the Niagara Cloud, describes license and software requirements, as well as procedures to configure the remote station for cloud connectivity and to register the device.

To use the Niagara Cloud, you must have a properly licensed Niagara Workbench PC. Additionally, you must have a JACE-8000 that is properly licensed for nCloudDriver. Internet access is required for all systems as well.

Overview

The nCloudDriver module contains the **NiagaraCloudNetwork** component and the functionality to send and receive data to and from the cloud. The main purpose of the module is to provide a mechanism to push data from a **NiagaraStation** (JACE or Supervisor station) to be stored, managed, and controlled from the cloud.

This module acts as an adapter, bridging Niagara's internal data to a cloud-specific format. It is modeled in such a way that one **NiagaraCloudNetwork** is all you should need, even if your configuration uses multiple cloud platform providers, such as AWS and Google. Each **CloudDevice** contains the same set of generic device extensions, which are capable of supporting each of the cloud platforms.

NOTE: At initial release, the nCloudDriver supports only the Honeywell Forge cloud platform, however, both the **CloudConnector** and **NiagaraCloudNetwork** are designed with extensibility in mind.

Once installed, the **CloudConnector** component provides device registration and a secure connection to the Honeywell Forge cloud platform. The Niagara Cloud then provides an application protocol to share data to the cloud.

The overall data flow occurs as follows: The station includes the **NiagaraCloudNetwork** and **CloudSentienceDevice**, which contains alarms, histories, points, system commands and events. This station also includes the **CloudConnector** component, which has an established connection to the cloud platform. Data are sent from the station to this cloud platform. As data enter the platform, some are stored in databases while other data are held in Event Hubs. Separately, there are cloud-based web applications, which consume the data, retrieving and reading data from either the Event Hub or REST APIs.

Requirements

This topic describes the licensing and software requirements for using the Niagara Cloud.

Platform requirements

The Niagara Cloud functions on any JACE-8000 or Supervisor platform.

NOTE: In Niagara, the Niagara Cloud is not supported for JACE-3E, JACE-6/6E, or JACE-7 platforms.

License requirements

To use the driver in a station, the nCloudDriver license feature must be enabled on the host. Additionally, the license feature requires limits to be set for device, points, and histories, otherwise those respective device extensions prevent operation.

The **CloudConnector** is also a licensed feature.

Also for any non-Forge Connect installation, you must be a registered user of the Asset Manager tool in Niagara Community to register a device for cloud communications.

Additional software requirements

- The software modules listed here must be installed on your system, followed by a station restart.
  - nCloudDriver-rt, -wb
  - cloudConfig-rt
  - cloudConnector-rt
  - cloudIotHubConnector-rt
  - cloudIotHubDep-rt
  - cloudSentienceConnector-rt, -ux

  For any Forge Connect installation, the following modules must be installed on your system as well.
  - fcModelSync
  - fcTagDict
  - fcEasyOnboard
- The Host Id for the device must also be added to the organization within the Asset Manager tool in Niagara Community.
- Internet access is required for all stations and clients.

  NOTE: If your local network uses a proxy server to provide Internet access, work with your on-site IT department to configure the proxy server to allow access to the Honeywell Forge platform. CloudLink(SM) requires that any intermediate proxy server be a fully transparent proxy; named proxy servers, also known as explicit proxy servers, are not supported at this time. The station requires unauthenticated proxy access to the following domains which are part of the Honeywell Forge ecosystem:
  - *.scm.azurewebsites.net
  - *.azurewebsites.net
  - *.cloud.tridium.com
  - *.tridium.com
  - *.dsentience.net
  - *.honeywell.com
  - *.force.com
  - *.niagara-community.com
  - *.trafficmanager.net
  - *.azure-devices.net
  - *.honeywellcloud.com
  - *.windows.net
- A Workbench connection to the remote station is required in order to configure the Niagara Cloud.

Installing software modules

This procedure describes downloading and installing required software modules and restarting the station on a PC.

Prerequisites:

- Internet connectivity
- An existing user account on the Niagara Community software portal.

  NOTE: A user account on the Niagara Community software portal is needed only for non-Forge Connect installations. Forge Connect installations do not use this software portal.

Step 1 In the web browser, log in to the Niagara Community software portal (`https://www.niagara-community.com`) and click **Software** (located upper right).

Step 2 Scroll down to locate the **`nCloud Honeywell Sentience Driver`** software and click on the appropriate .zip file link depending on your software release version. For example, for Niagara 4.9 the .zip file is: `nCloud_Honeywell_Sentience_Driver-4.9.0.2.zip`

The zip file downloads to your system.

Step 3 In your Windows user downloads folder (`C:\Users\[UserName]\Downloads`), right-click on the .zip file and extract the contents to your SysHome installation folder (`Niagara/Niagara4 x.x`).

NOTE: Install only Niagara Cloud-4.9.0.2 on Niagara 4.9or higher

Step 4 If the system displays a prompt to `Overwrite any existing previous versions?`, click **OK**.

The nCloudDriver software modules and palettes are ready to use.

Step 5 Restart the station.

Once the station restart is complete, you can proceed to install and configure the **CloudConnector**, **NiagaraCloudNetwork**, and **CloudSentienceDevice**.

Setting up device internet access

Internet access is required for all stations and clients. If your device is on an internal (closed) network, this is done by setting up proxy server settings, typically handled by the on-site IT department. Additionally, you must enter the proper DNS Server and DNS Host configuration for your network on the device.

Prerequisites: A platform connection has been opened to the remote device. For a device behind a network firewall, the appropriate DNS Host Name and DNS Server IP address(es) for your network are defined.

NOTE: If your local network uses a proxy server to provide Internet access, work with your on-site IT department to configure the proxy server to allow access to the Niagara Cloud platform. The Cloud Connector requires that any intermediate proxy server be a fully transparent proxy; named proxy servers, also known as explicit proxy servers, are not supported at this time. The station requires unauthenticated proxy access to the following domains, which are part of the Niagara Cloud ecosystem:

- *.scm.azurewebsites.net
- *.azurewebsites.net
- *.cloud.tridium.com
- *.tridium.com
- *.dsentience.net
- *.honeywell.com
- *.force.com
- *.niagara-community.com
- *.trafficmanager.net
- *.azure-devices.net
- *.honeywellcloud.com
- *.windows.net

Step 1 In the platform **TCP/IP Configuration** view, enter the appropriate values for the following properties:

- **`DNS Domain`** (ex: company.net)
- **`DNSv4Servers`** (add a field for one or more DNS Servers, enter the appropriate IP address for each)

Step 2 Click **Save**.

Upon saving your changes you are prompted to reboot the device.

CAUTION: From a cyber security perspective, it is crucial that your station is not exposed to the Internet. The **CloudConnector** component requires an outbound connection from your station to the Internet, but should not allow incoming connections to your stations from the Internet. For this reason, we strongly recommend following the best practices and steps in the *Niagara 4 Hardening Guide*, which is available on: https://www.tridium.com/us/en/services-support/library.

Chapter 2 Install and configure

Topics covered in this chapter

- Cloud connector
- Adding the cloud connector service
- Registering the device
- Adding the NiagaraCloudNetwork
- Adding a CloudSentienceDevice
- Configuring the station to receive commands
- Configuring Role Mappings
- Using the Cloud Event Recipient

If you are using Forge Connect, the **ForgeConnect EasyOnboardService** installs and configures the **CloudConnector** for you. The procedures in this section apply only to non-Forge Connect installations.

For non-Forge Connect installations. these procedures describe the steps to add the **CloudConnector** service to your station, register the device with the Asset Manager tool, as well as add the **NiagaraCloudNetwork** and **CloudSentienceDevice**.

Most components required for communicating to the cloud are under the **NiagaraCloudNetwork**. This network provides logical organization for many types of clouds.

Cloud connector

A cloud connector component manages communication between a station and the cloud.

NOTE: Cloud connector is a licensed feature.

A cloud connector component provides connectivity to a specific cloud platform. For example, **CloudConnector_Sentience** connects with the Honeywell Forge platform. You may create several cloud connectors, each configured to connect to a different cloud platform, for example, Amazon Web Services (AWS), Microsoft Azure, Bluemix, Cloud Foundry, Google Cloud.

Once a cloud connection is established, the cloud connector maintains the connection, sending messages (that is, heartbeats), handling device registration operations, and reporting on connectivity and device registration status.

Adding the cloud connector service

You must add the cloud connector service to stations that do not support Forge Connect. This procedure documents how to add this service to the **Services** container in a Workbench Nav tree.

Prerequisites: You are working in Workbench connected to a station. You are using a branded **`nCloudDriver`** palette.

It is important to get the correct cloud connector component installed in your station. This pluggable **ConnectorImpl** component configures the connection to Honeywell Sentience.

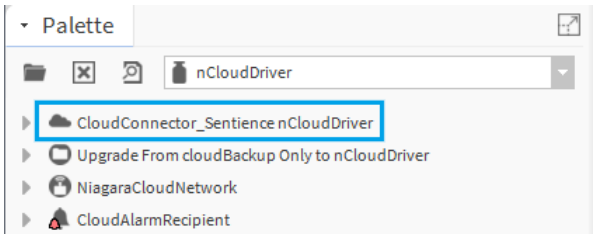
Step 1 To open the **Palette** side bar, select **Window→Side Bars→Palette** from the **Menu** bar.

The **Palette** side bar opens.

Step 2 Click the **Open Palette** folder icon ( ).

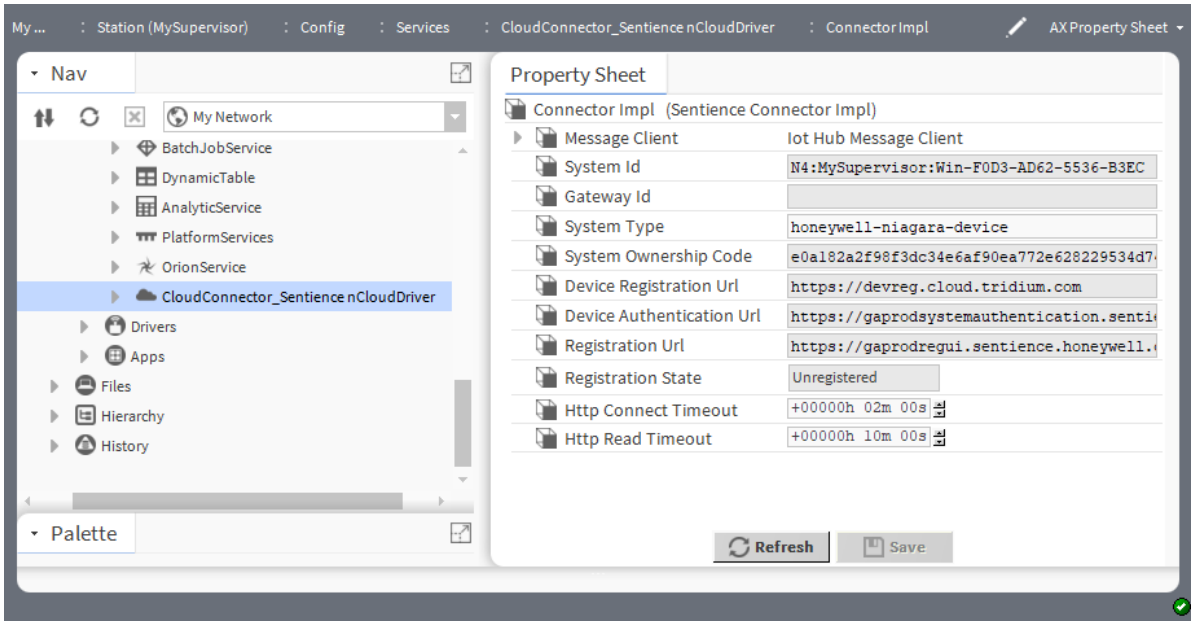
The **Open Palette** window opens.

Step 3 To add the cloud connector component, select the **`nCloudDriver`** palette, click **Open** and drag the **CloudConnector_Sentience nCloudDriver** component to the **Services** folder in the Nav tree.



Step 4 Right-click the component you just added, select **Views→AX Property Sheet** and click **Connector Impl**.

The **Sentience Connecter Impl Property Sheet** opens.



`System Type`s are specifically registered with the Honeywell Forge Identity Provider, and only defined **`System Type`**s can be used for device registration.

If you are using a branded **`nCloudDriver`** palette, **`System Type`** is pre-configured by the palette. If you are not using a branded **`nCloudDriver`** palette, **`System Type`** defaults to blank. This property is key to getting the device correctly registered and must not be blank. Although it is possible to manually enter the **`System Type`**, a best practice is to use the external palette provided in the software image download. This software image contains one palette with the **`System Type`** already correctly configured for your brand.

Step 5 If **`System Type`** is blank, enter the correct type and click **Save.**

You can obtain the correct system type from your support channel.

Once the component is in place, the next step is to register this device with the Asset Manager in Niagara Community and Honeywell Forge.

Registering the device

For non-Forge Connect installations, this procedure describes how to register the cloud connector installed on your device with the Asset Manager tool in Niagara Community and the Honeywell Forge platform. Initiating device registration from the **CloudConnector** provides the connection information required to connect the device securely to the cloud platform.

Prerequisites:

NOTE: For Forge Connect installations, users access the Honeywell Cloud Admin Portal for device registration during the Forge Connect EasyOnboard process.

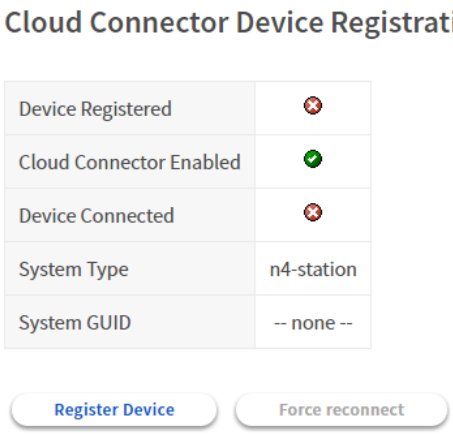
Prerequisites listed here apply to non-Forge Connect installations only.

- The cloud connector component is installed in station's **Services** container.
- A user account for Niagara Community exists.
- A user account for the Asset Manager tool exists within the Niagara Community.

Step 1 If you do not have an existing account in the Asset Manager tool, click the **Getting Started** link in Niagara Community and sign up for one.

Step 2 Expand the station **Services** container and double-click on the cloud connector.

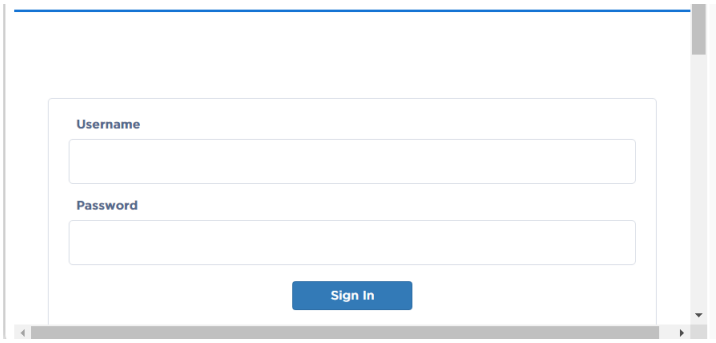
The **Cloud Connector Device Registration** view opens.



Step 3 Click **Register Device**.

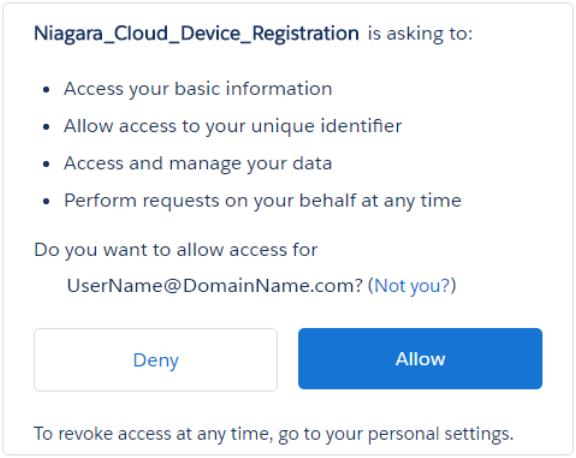
For Forge Connect installations, the **`System Type`** is: `honeywell-niagara-device`.

The **Sign In** view opens.



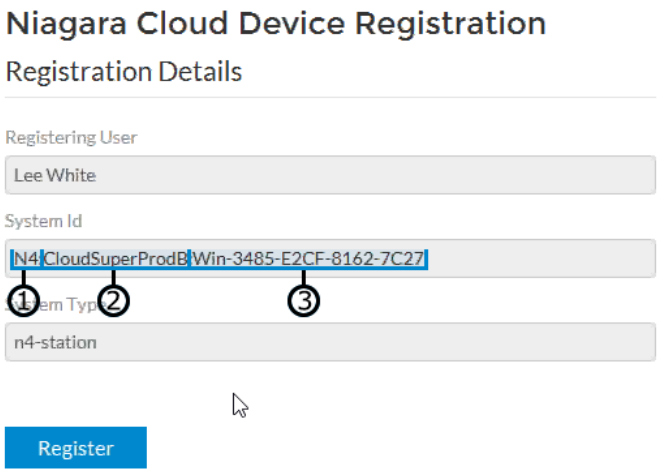
Step 4 Enter your user credentials for the Asset Manager tool and click **Sign In**.

The Niagara_Cloud_Device_Registration view opens.



Step 5 To grant the registration site access to your device and user information, click **Allow**.

The **Niagara Cloud Device Registration—Registration Details** view opens.



This view displays your registration details. The **`System Id`** in the **Registration Details** (shown above) includes the following information:

1	Platform version. For example: N4
2	Station name. For example: CloudSuperProdB
3	Host Id. For example: Win-3485–E2CF-8162–7C27

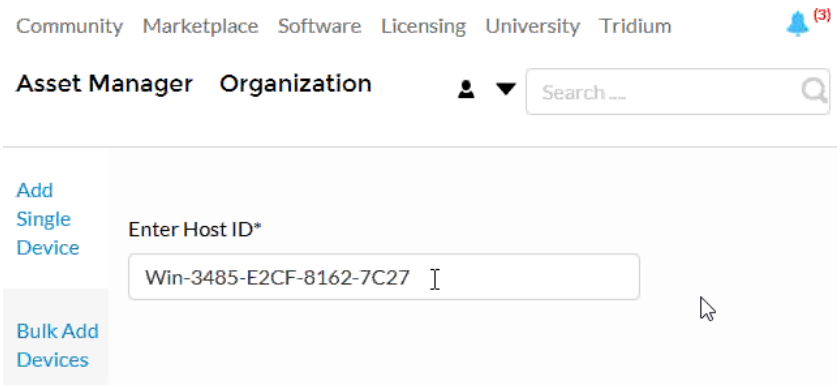
`System Type` is pre-configured by the cloud connector.

Step 6 To proceed, click **Register**.

CAUTION: Renaming your station after it is registered changes the **`System Id`**. Since the station name is part of the point Id and the backup identity, changing the station name after registration creates a disconnect between cloud data collected before and after the rename occurred. To change the station name, you must re-register the station.

Step 7 If the certificate **Identity Verification** window opens, accept the site certificate by clicking **Accept**.

The **Add Single Device** view opens and displays the actual Host ID for your controller

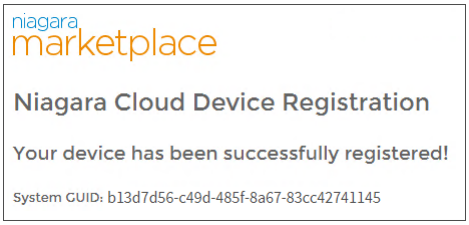


NOTE: Do not change this Host ID value since it matches your platform's Host ID.

Step 8 To proceed, click **Continue**.

Step 9 Add additional details for your device and your organization, and click **OK**.

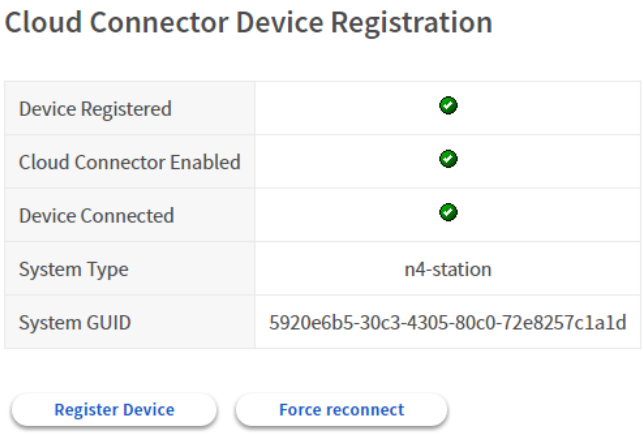
The **Add Single Device** view displays a confirmation message saying the device is successfully added to the asset manager and an additional confirmation message displays.



The device is now registered with Honeywell Forge and can send data to the cloud.

Step 10 Click **Done**.

The **Niagara Cloud Device Registration** view displays three white check marks in three green circles.



Adding the NiagaraCloudNetwork

The NiagaraCloudNetwork is a container for most components required to communicate with the cloud platform. Specifically, this includes the **CloudSentienceDevice**, which is configured for the cloud connector and contains the appropriate protocol for connecting to and communicating with the cloud platform. This

procedure describes the steps to add the **NiagaraCloudNetwork** component under the station **Drivers** container.

Prerequisites: You are working in Workbench connected to a station.

Step 1 Open the **nCloudDriver** palette.

Step 2 In the palette, drag the **NiagaraCloudNetwork** component and to the station's **Drivers** container.

The **Name** window opens.

Step 3 Enter a name for the network (or leave the name at its default) and click **OK**.

Adding a CloudSentienceDevice

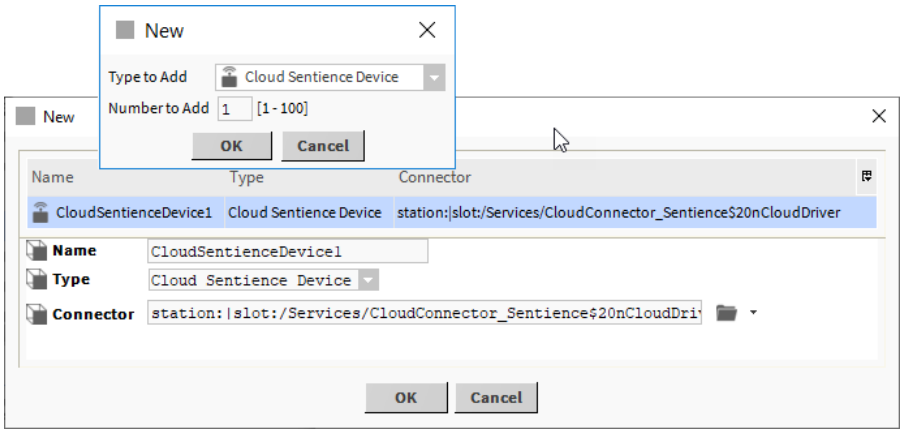
Use the following procedure to add a new **CloudSentienceDevice** to the **NiagaraCloudNetwork**.

Prerequisites: You added the **NiagaraCloudNetwork** to the station's **Drivers** folder.

Step 1 Double-click the **NiagaraCloudNetwork**.

The **Cloud Device Manager** opens.

Step 2 Click **New** and select `Cloud Sentience Device` from the **Type to Add** drop down list and click **OK**.



Step 3 In the second **New** window, name the device as desired.

Step 4 Accept the default ORD or expand the folder icon ( ) to the right of the **Connector** property and use the **Component Chooser** to select the cloud connector and click **OK**.

Make sure that the Connector property (an ORD) points to a properly configured and connected cloud connector. When you add the device using the **New** button the driver automatically selects the connector if there is only one, and, typically there is no need to have more than one connector installed in a station.

The driver adds the **CloudSentienceDevice** to the network device where it appears as a row in the **Cloud Device Manager** table.

Configuring the station to receive commands

Configuring a station to receive commands requires adding the **CloudAuthenticationScheme** and at least one or more **CertTrustMapping** or **JwksTrustMapping** components.

For successful configuration, you need to have at least one **CertTrustMapping** or **JwksTrustMapping** component. To enable your station to accept tokens from all desired token providers, you can have more than one and any combination of these components.

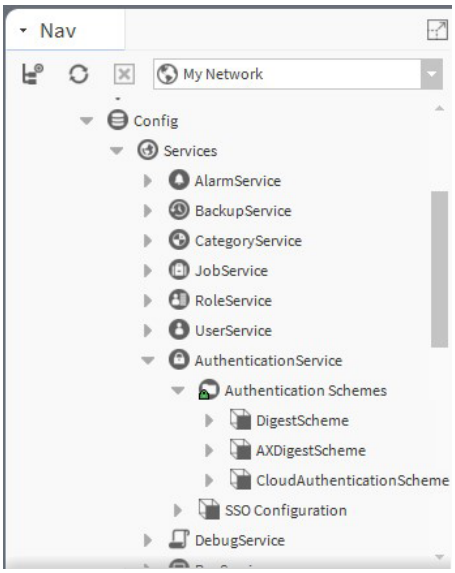
Adding the CloudAuthenticationScheme

This procedure describes how to add a **CloudAuthenticationScheme** to your station.

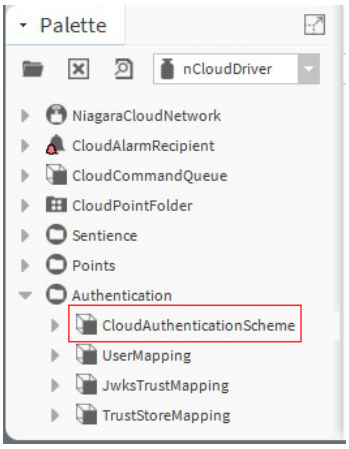
Prerequisites:

- The **nCloudDriver** palette is open.

Step 1 Expand **Config→Services→Authentication Service→Authentication Schemes**.



Step 2 Expand the **Authentication** folder in the palette, drag the **CloudAuthenticationScheme** component to the **Authentication Schemes** folder in the Nav pane.



The **CloudAuthenticationScheme** is now in your station and available for adding **CertTrustMapping** or **JwksTrustMapping**.

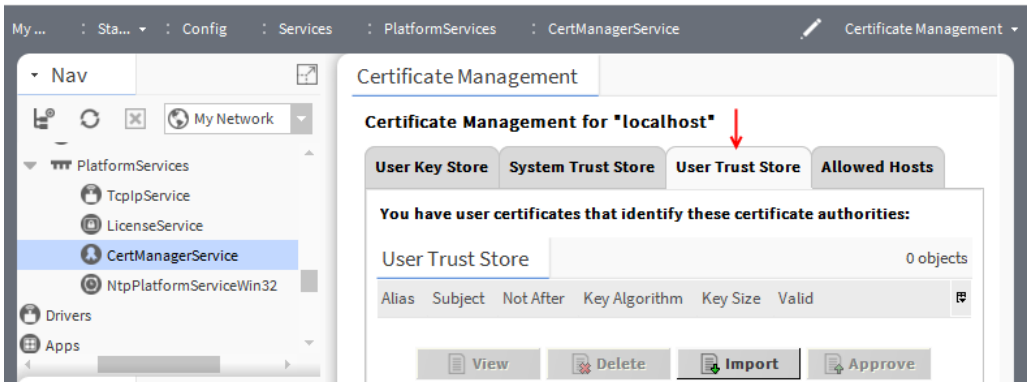
Adding a CertTrustMapping

This procedure describes how to add **CertTrustMapping** to a station.

Prerequisites: You have the public certificate of your token provider and know the value for the Token issuer "iss" property (provided by the Developer/Integrator during Certificate Trust Mapping configuration). A **CloudAuthenticationScheme** has been installed.

Step 1 Expand **Config→Services→PlatformServices** and double-click **CertManagerService**.

The **Certificate Management for "localhost"** opens.



Step 2 Select the **User Trust Store** tab and click **Import**.

The **Certificate Import** window opens.

Step 3 Browse to the location of your token provider's public certificate file, select the file and click **Open**.

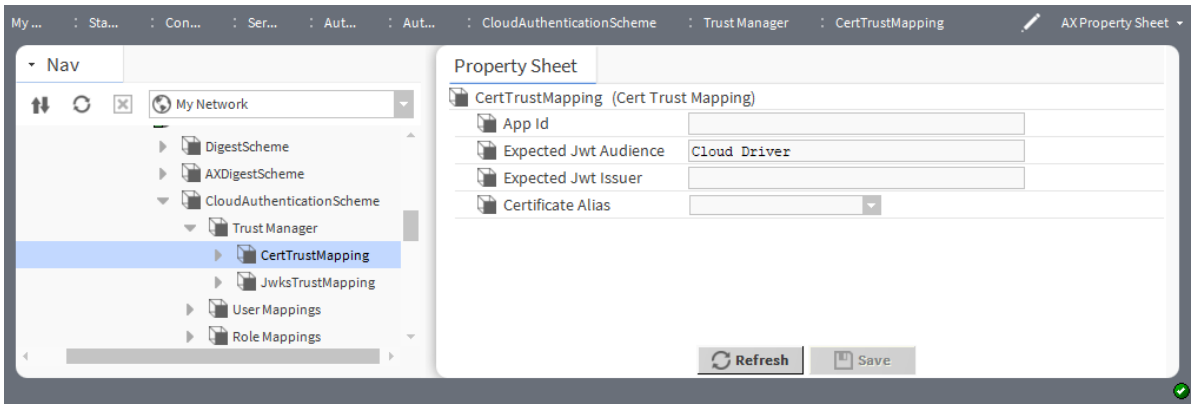
Step 4 Enter an alias for the certificate and click **OK**.

Step 5 In the Nav tree, expand the **CloudAuthenticationScheme** component (added in the previous topic).

Step 6 From the **nCloudDriver** palette, drag a **CertTrustMapping** component to the **Trust Manager** node under the **CloudAuthenticationScheme**.

Step 7 Right-click the new **CertTrustMapping** component and select **Views→AX Property Sheet**.

The **Cert Trust Mapping Property Sheet** opens.



Step 8 In the **App Id** property, enter the Honeywell Forge application ID.

This value is required.

Step 9 In the **Expected Jwt Issuer** property, enter the value of the token issuer "iss".

This is typically the URL of the user identity provider.

Step 10 In the **Expected Jwt Audience** property, enter the value of the token audience "aud".

By default, this is "Cloud Driver", but may be changed to match the value present in the JWT for those providers that do not have a fully configurable audience property. For example, Salesforce prepends the Salesforce application ID (not to be confused with the Honeywell Forge application ID) onto the audience.

Step 11 In the `Certificate Alias` property, expand the drop-down and select the alias of the certificate that was imported above.

Step 12 Save the **CertTrustMapping** component configuration.

Certificate trust mapping is now available and the station is configured to receive commands.

Adding a JwksTrustMapping

This procedure describes how to add **JwksTrustMapping** to a station.

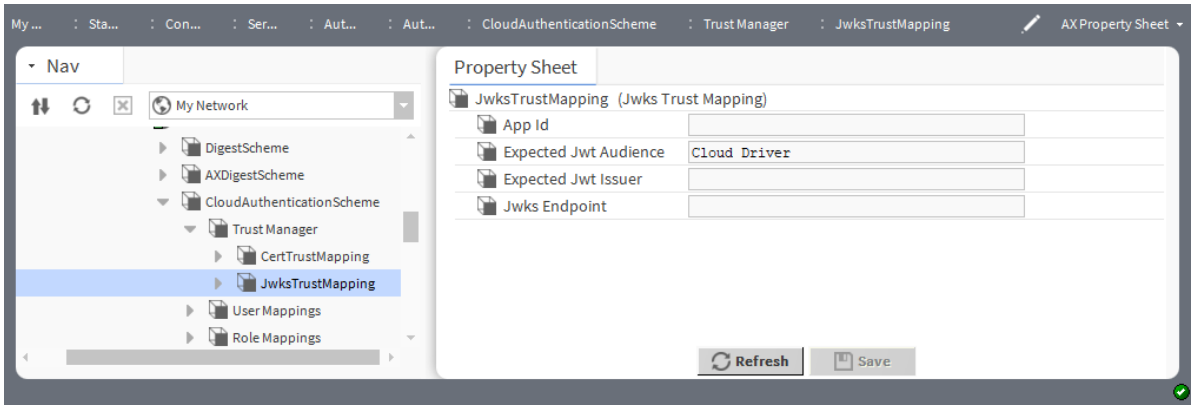
Prerequisites: You know URL of the token issuers key service. You know the value for the Token issuer "iss" property (provided by the Developer/Integrator during JwksTrustMapping configuration).The **CloudAuthenticationScheme** has been installed.

Step 1 In the Nav pane, under the **AuthenticationService**, expand **CloudAuthenticationSchemes→CloudAuthenticationScheme→TrustManager**.

Step 2 From the **`nCloudDriver`** palette drag a **JwksTrustMapping** component to the **Trust Manager** node in the **CloudAuthenticationScheme** in the Nav tree.

Step 3 Right-click the newly added **JwksTrustMapping** component and click **Views→AX Property Sheet**.

The **Jwks Trust Mapping Property Sheet** opens.



Step 4 In the **`App Id`** property, enter the Honeywell Forge application ID.

This value is required.

Step 5 In the **`Expected Jwt Audience`** property, enter the value of the Token audience "aud" property.

By default, this value is "Cloud Driver", but you may change it to match the value present in the JWT for those providers that do not have a fully configurable audience property. For example, Salesforce prepends the Salesforce application ID (not to be confused with the Honeywell Forge application ID) onto the audience.

Step 6 In the **`Expected Jwt Issuer`** property, enter the value of the Token issuer "iss".

This is typically the URL of the user identity provider.

Step 7 In the **`Jwks Endpoint`** property, enter the URL of the token issuer's key service.

Step 8 To save the **JwksTrustMapping Property Sheet** settings, click **Save**.

Jwks trust mapping is now available and the station is configured to receive commands.

Configuring Role Mappings

This procedure describes how to configure roles, which authorize System Commands to access station resources. You need one role mapping for each cloud role contained in your security token. You can map more than one cloud role to the same station role, if necessary.

Prerequisites: The **CloudAuthenticationScheme** and **JwksTrustMapping** are already configured. The **nCloudDriver** palette is open.

Step 1 In the **nCloudDriver** palette, expand **Authentication**.

Step 2 Drag a **Role Mapping** component from the palette to the **Role Mappings** component under **CloudAuthenticationScheme→Role Mappings** and double-click the **Role Mapping** component.

Step 3 For the **Cloud Role** property, enter the exact name of one of the cloud role that will be in the claim of your security token.

Step 4 For the **Station Role**, enter the exact name of an existing role in the **RoleService** of the station.

NOTE: Do not enter the default "Admin" role for the station role. The driver ignores any role mapping with a station role of "Admin" for security reasons.

Step 5 Click **Save**.

The station is now ready to receive commands with cloud roles specified.

For more details, see .

Using the Cloud Event Recipient

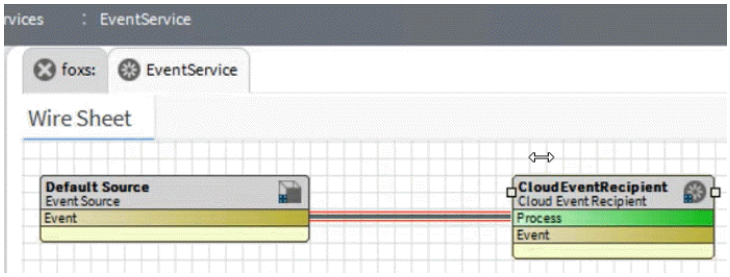
In this procedure you configure the component to receive event data from Niagara as it is generated and send it to the cloud platform.

Prerequisites: A configured **NiagaraCloudNetwork** object must exist in the station for the **CloudEventRecipient** to function properly. Additionally, the **CloudConnector** must be properly configured in the station's **Services** folder.

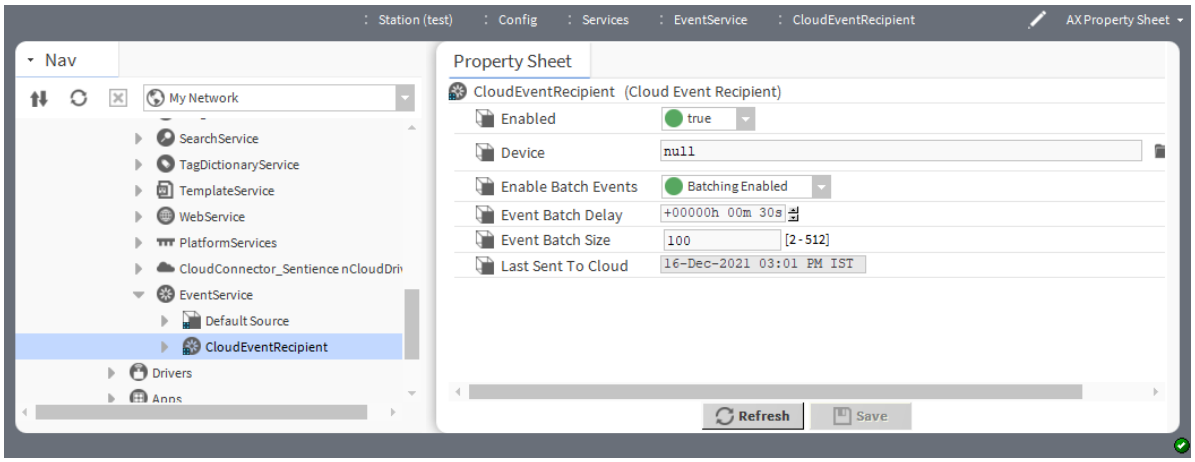
Step 1 If your station does not have an **EventService**, open the **event** palette and add an **EventService** to your **Services** folder.

Step 2 Navigate to the station **Services→EventService**, and open the **Wire Sheet** view.

Step 3 From the **nCloudDriver** palette, add the **CloudEventRecipient** component to the **EventService Wire Sheet**.



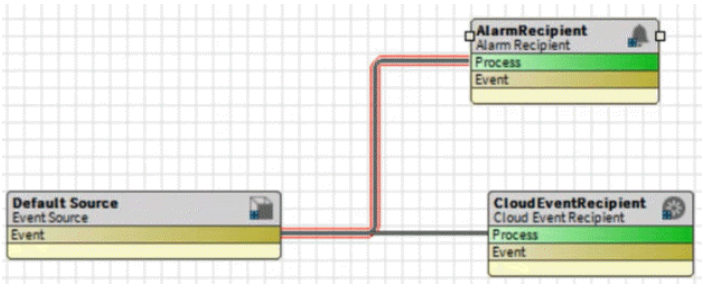
Step 4 Open a **Property Sheet** view of the component, and configure the **Device** property with the **CloudSentienceDevice** located under the **NiagaraCloudNetwork**.



Step 5 In the **EventService Wire Sheet**, click to link the **`Event`** slot of the **DefaultSource** component to the **`Process`** slot of the **CloudEventRecipient**.

This ensures that events are routed to the **CloudEventRecipient** appropriately.

Optionally, you can add an **Alarm Recipient** component from the event palette to the **Wire Sheet** and link the **`Event`** slot of the **DefaultSource** component to the **`Process`** slot of the **Alarm Recipient** to route events to the alarm database. This gives you a record of the event even after the event data are sent to the cloud.



Chapter 3 From station to the cloud

Topics covered in this chapter

- ♦ Application design considerations
- ♦ Configuring point data for the cloud using Cloud Point Learn
- ♦ Configuring tuning policy
- ♦ Configuring history data for the cloud using History Learn
- ♦ Preparing alarms to send to the cloud
- ♦ Performance optimization

The nCloudDriver manages two-way traffic, from a station to the cloud, and from the cloud to a station. Going from a station to the cloud pushes locally stored data to a central location in the cloud where it can be combined with data from other locations.

Most of the following topics describe communications that are sent from a station to the cloud. These topics include important information about how to optimally design, configure, and tune your station's point, history and alarm data when communicating with Honeywell Sentience cloud.

Application design considerations

When setting up a station database and designing a connected application to interact with a station via Sentience, consider carefully the type of data you want to export to Sentience, and its expected use.

Points versus Histories

Cloud proxy points and cloud history exports use the same Sentience message structure to send data to the cloud. The two mechanisms push data into the same database. Therefore, you should choose whether to export station data using histories or points based upon the nature of the data and its usage. Differences between the way point and history information is collected and sent to Sentience may influence your design.

Cloud proxy points use a batch snapshot approach to export point data. At assigned intervals, three batch update threads collect a current snapshot of each source point's value and status. They add these data to the list of batch updates, which are subject to tuning rules that determine if a point update should be sent. At last, they send these values to Sentience. If the update interval is 5 minutes, the driver sends the value once every 5 minutes. If the point experiences a value change in between update intervals, an update is not reported, only the value at the 5 minute interval is sent. This is analogous to an interval history extension in Niagara.

Cloud proxy points can be put into a COV (change of value) mode, where the driver sends every point value and status change to Sentience, which is subject to change tolerance (refer to "Configuring Tuning Policy"). However, this is inefficient. Instead of sending thousands of points per message as is possible with batch updates, the driver sends every point change as an individual message. Due to limitations with the Sentience IoT Hub, it is recommended to not use more than 20 COV points at a time, and only for debug and diagnostic purposes. It is not recommended to use COV as a normal operation mechanism.

NOTE: Retrieving real-time data for Sentience Connected Application developers: To obtain point values, use the **CloudMultiPointReadCommand** to request multiple point values. For pushing values,use the **CloudMultiPointWriteCommand** to send multiple values to the station at once.

CloudHistoryExport can export all histories in the database, including histories imported from other stations (for example, using drivers like BACnet, OBIX, and NiagaraDriver) to Sentience. The history extension that generates the records or imports them from the foreign device governs the collection frequency for the history. **CloudHistoryExport** retrieves all records that have not yet been sent for a history, and includes them in the update message sent to Sentience.

NOTE: Capturing every point change: The best way to push every point change to Sentience without missing any value changes would be to add a Niagara **CovHistoryExt** to the point, so that every point change is captured to a Niagara history. Then use a **CloudHistoryExport** to export the history records in bulk.

It is important to note that JACE platforms have a limit to the number of histories they can contain. This is an inherent OS limit due to the number of allowed file descriptors, one of which is consumed by each history. While the exact number you might be able to reach in an individual station depends on several variables, the general limit is no more than 6000 histories included in a history database resident on a JACE-8000. This limit does not apply to Windows-based Supervisor stations, nor to Linux-based Supervisor. There is no defined limit on the maximum number of cloud proxy points, although you are still subject to space limitations and to driver licensing restrictions.

Configuring point data for the cloud using Cloud Point Learn

Adding proxy points to the NCHSD for large stations (with more than ten thousand points) uses point learn rather than discovery. The point learn process minimizes the effect on performance and organizes the large number of points into a logical structure based on folders in the station.

Prerequisites: Cloud connectivity and communication are already configured for the station. The source station contains points to be exported.

Exporting cloud proxy points (that is, pushing point data to the cloud) is similar to point discovery in other drivers. The key difference is that for other drivers you select data to be brought into the **NiagaraStation**, whereas for this driver, you select **NiagaraStation** data to send out of the station to the Honeywell Forge cloud.

In most cases, you want to export all (or nearly all) of a station points to Sentience. The point learn feature allows you to automatically add all the points to the cloud export list in a way that bypasses much of the issues that cause delays during regular point discovery. To avoid putting all the proxy points into a single flat folder, which would cause significant problems when trying to open or view the folder, a point learn replicates the folder structure that it finds in the station using **CloudPointFolder**s inside the **CloudPointDeviceExt**.

Step 1 Expand **Config→Drivers→NiagaraCloudNetwork→CloudSentienceDevice** and double-click **Points**

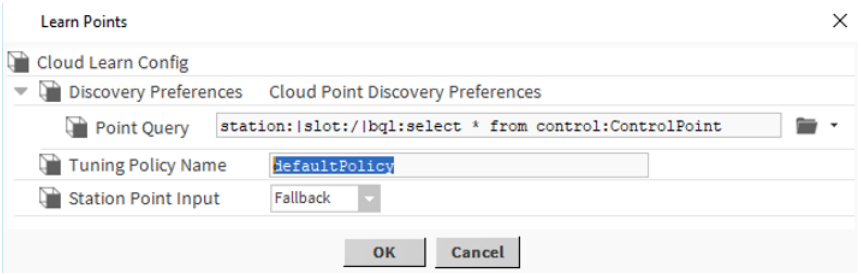
The **Cloud Point Manager** view of this component opens.

Two buttons for point discovery are available: **Discover** and **Point Learn**. For large stations, using the **Discover** button may cause you to experience very long wait times in both the discovery and the add phases. As the station gets beyond 10,000 points, this becomes significantly longer, and eventually will prevent a successful discovery job without modifications to the system properties for managing the fox connection between the Workbench and station.

Step 2 Do one of the following:

- To use a discovery job for a small number of points, click **Discover** at the bottom of the view. When the **Discovered** table displays the points, select them and click **Add**.
- To use automatic point discovery in a large installation, click **Point Learn** at the bottom of the view.

When you click **Point Learn**, the **Learn Points** window opens.



The point learn configuration involves two parts: configuration of the discovery points and configuration of the cloud proxy points.

- The discovery of points in the station using point learn is the same as regular point discovery using the **Discover** button on the **Cloud Point Manager** view. The **`Discovery Preferences`**' **`Point Query`** is a BQL or a NEQL query that selects the points from the local station.

  NOTE: If the query is configured to return non-control point results, the points will not properly export to the cloud.

- Proxy points in the cloud involve specifying two properties: **`Tuning Policy Name`** and **`Station Point Input`**. These properties apply to the cloud proxy extension added for each discovered point.

You can limit the query to a particular part of the station database or to a type of point. You can also use **`Tuning Policy Name`** and **`Station Point Input`** to apply this setting to each added proxy point.

You can perform repeated point learn invocations without conflict. Points that already exist are not to be duplicated or added again. This also means that you could learn different sections of the station database in chunks, if you prefer to make the construction of the necessary BQL queries easier.

Step 3 Expand **`Discovery Preferences`**, configure the properties including the **`Point Query`** and click **OK**.

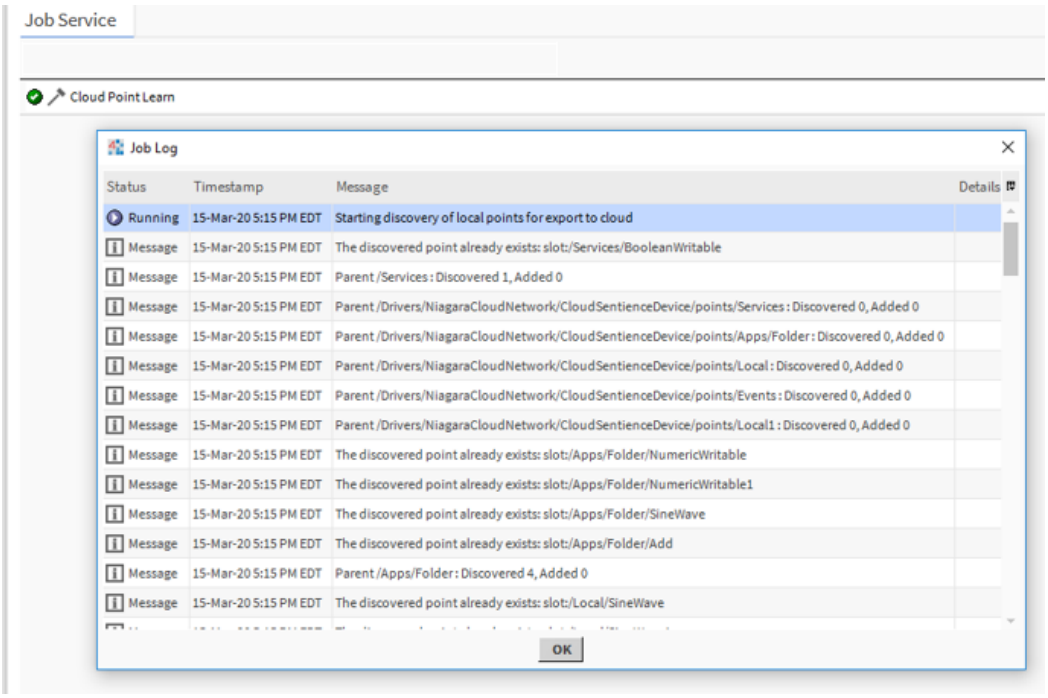
The points that are identified by the configured **`Point Query`** are added to the station. The folder structure of the station itself is replicated as needed in **CloudPointFolders**, so that the same organizational hierarchies created for the source points are present for the cloud proxy points. This makes it much easier to find cloud proxy points, and it makes the discovery process and subsequent management of the proxy points much more performing.

Step 4 To view the **Job Log** that displays job results, expand **Config→Services** and double-click **JobService**.

The **Job Service Manager** opens.

Step 5 Click on the double chevron (>>) icon to the right of the **Cloud Point Learn** job to view the log.

The **Job Log** opens.



Configuring tuning policy

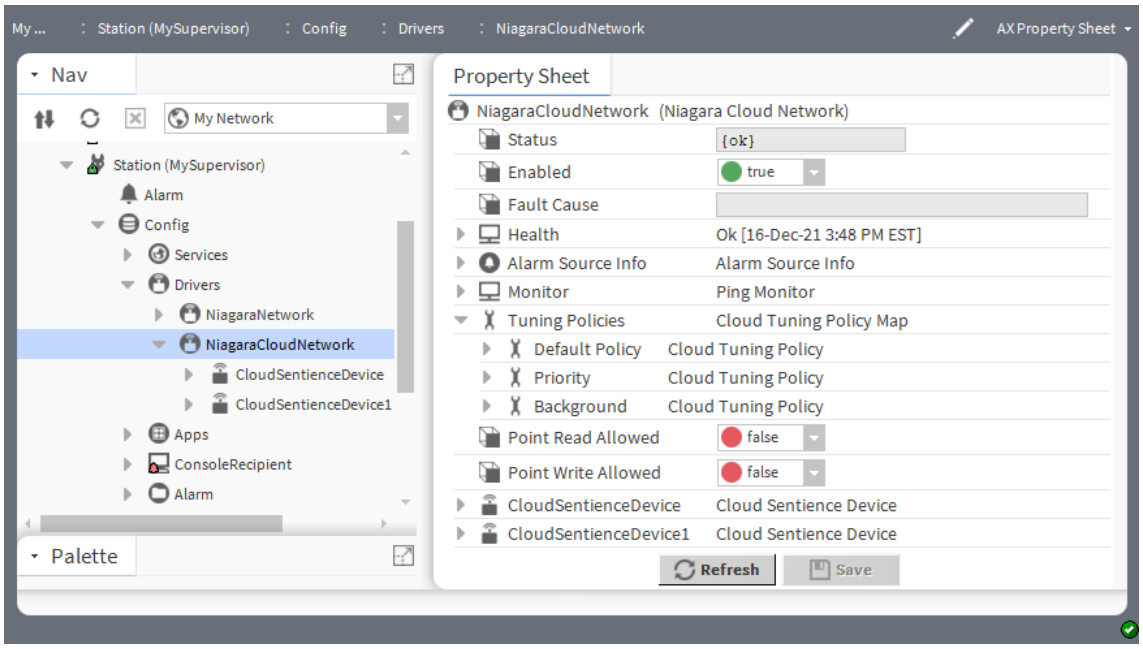
The cloud connector uses tuning policies to govern the update of point values to Honeywell Sentience. These policies provide a variety of configuration options for configuring point updates to the cloud.

Prerequisites: You are using Workbench and are connected to a station.

By changing a tuning policy, you are changing the configuration for all points that reference this policy. This makes it convenient to specify different policies for different groups of similar points, which can then be configured together.

Step 1 To locate the proxy point tuning policies, expand **Config→Drivers**, right-click **NiagaraCloudNetwork**, select **Views→AX Property Sheet** and expand **Tuning Policies**.

The **Niagara Cloud Network Property Sheet** opens with the **Cloud Tuning Policy Map** expanded.

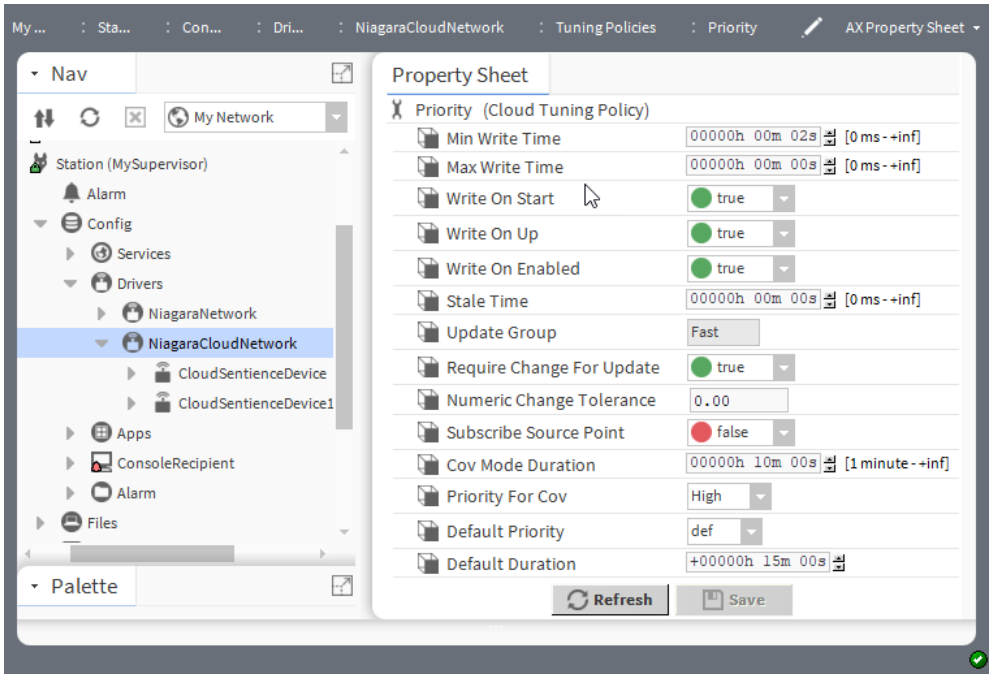


Three default policies are available:

- Default Policy
- Priority
- Background

Step 2 To modify a tuning policy, expand or click one of the policies.

The policy properties open.



The screen capture shows the default tuning policy as it is configured out-of-the-box.

- **Require Change for Update** defaults to `true`. If the value of the point has not changed, the driver excludes the point from the batch point update (learn point). This helps to minimize the amount of data sent. To send the point value even if it has not changed, set this to `false`, but be aware that this may vastly increase the traffic sent to Honeywell Sentience from your station.
- **Numeric Change Tolerance** may help in configuring numeric points for batch update. This property includes only significant value changes as defined by the tolerance value.
- **Subscribe Source Point** configures the freshness of subscribed points. If you export points that are integrated through another driver such as BACnet, those points may not be subscribed at all times. Points that exist to be used on graphics are only subscribed when the graphic is being viewed. At other times, their value is stale, as indicated by their **Status**. To always export to Honeywell Sentience points with fresh values, set this property to `true`. Permanently subscribing a large number of points by setting this property to `true` may significantly impact driver performance. Many drivers use a lower-bandwidth bus like twisted-pair EIA-485, which may not be able to handle the larger number of messages required to update all the integrated points.

Step 3 Configure tuning policy properties and click **Save**.

Configuring history data for the cloud using History Learn

Exporting cloud histories (that is, pushing history data to the cloud) occurs much like other history exports. A history learn automatically discovers and adds each history in the station's history database to cloud history. History Learn is strongly recommended over manual discovery both for speed gains and to avoid inadvertent creation of duplicate history exports. History Learn does not provide a query filter option.

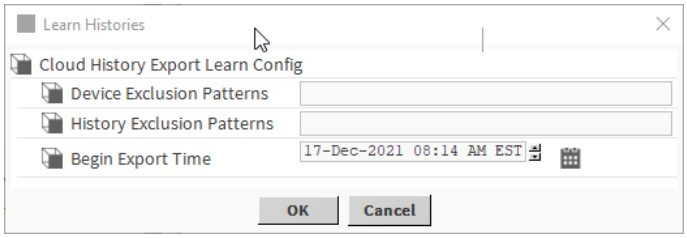
Prerequisites: Cloud connectivity and communication are already configured for the station. The source station contains histories to be exported.

Step 1 Expand **Config→Drivers→NiagaraCloudNetwork→CloudSentienceDevice** and double-click **Histories**.

The **Cloud History Export Manager** opens.

Step 2 Click **History Learn.**

The **Learn Histories** configuration window opens.



Step 3 Configure as needed any **Device Exclusion Pattern**, **History Exclusion Pattern**, **Begin Export Time**, and click **OK**.

The **Confirm** window opens.

Step 4 To add cloud history exports to all discovered histories, click **OK** (or you can cancel the export).

The history export learn job immediately discovers and adds all histories to the station's history database. This includes certain history types that may not be able to successfully export to Honeywell Sentience. For example, histories generated by the **System Monitor Service** cannot be exported to the cloud, but they are added to the station. You can delete these history exports manually after performing the history learn. You can also manually delete any other cloud history exports you do not wish to send to Sentience.

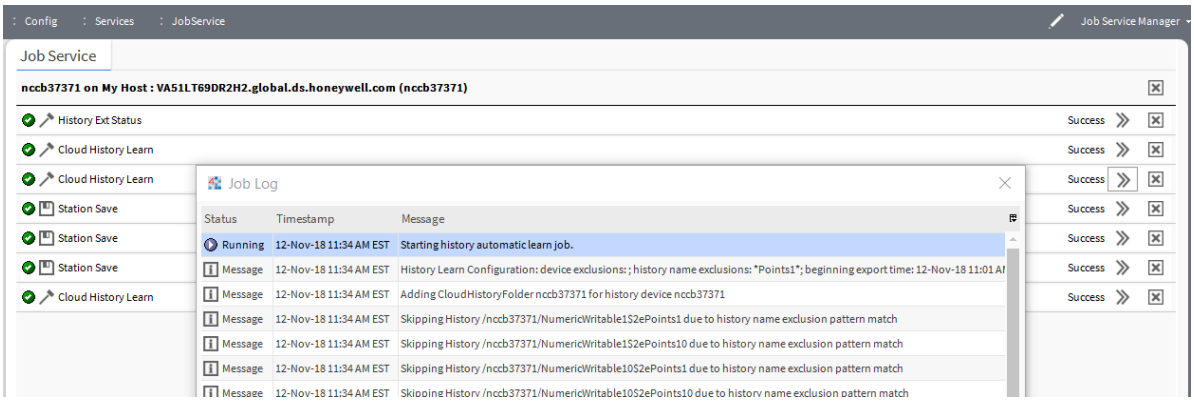
When the next history **Batch Trigger Time** occurs, the driver sends history records created after the **Begin Export Time** to the cloud. It does not send history records recorded prior to the

specified time. Once the driver learns the histories and starts to send them to the cloud, it continues to send to the cloud upon each history **`Batch Trigger Time`**. (You configure **`Batch Trigger Time`** on the cloud history device extension.)

The driver groups history exports into **Cloud History Folder**s replicating the **History Device** in which they are contained in the history database.

Step 5 To view the **Job Log**, expand the station's **Services** node and double-click **JobService**.

The **Job Service Manager** opens.



Step 6 To view the log, click the chevron (>>) icon to the right of the **Cloud History Learn** job.

The **Job Log** opens.

Discovery and export of very large history databases

Avoid problems when performing History Learn when history database contains extremely large number of records.

Currently the History Auto Learn capability will discover all the Histories in the station, and create a **`Cloud History Export`** for each history. This behavior is invoked by clicking the "History Learn" button in the **Cloud History Export Manager** view. While much faster than performing discovery and individually selecting histories for export, it can lead to problems if the history database contains an extremely large number of records, such as in a station that has several years' data. The learn process is not yet configurable, and attempting to archive several years' data, such as will happen on the first archival after the learn, may result in this station, or other systems, being blocked from communicating with Sentience. A future release of NCHSD will provide additional configuration capability so that systems can be integrated into Sentience without problems.

For integrating systems using the NCHSD GA release, please follow these guidelines for history archival:

1. The best option is to start with a new station, to avoid large history upload.
2. If you cannot use a new station, we recommend deleting the histories prior to exporting with NCHSD. You can do this from the **History Db Maintenance** tool. Select all the history devices, then select "**Delete All Records**" (or "**Delete Histories**" if using the **AX History Db Maintenance** view).
3. If you cannot delete the entire histories, try removing all records before a certain time. If there is a point in time before which the data can be ignored, this may help reduce the size of the database to be exported.

Iterative history export

This is an advanced procedure to try if it is absolutely not possible to accept loss of history data from a station to which you are adding NCHSD, you can potentially use this iterative process to get all the history records to the cloud.

Prerequisites:

- Before using this, please delete or clear records from any history that are not essential. Also, clear records before the earliest time you actually need the data. Any reduction in size you can achieve will make this process easier.

NOTE: For a very large database, the following may need to be done in many steps. Start with very small steps, perhaps even one history, and conservatively increase the number of histories you add. This is particularly important when the time range is long, as that means each history will have many records.

Step 1 Set the **`CloudHistoryDeviceExt Batch Trigger Time`** property to Manual.

This will prevent any archival attempts when you are not expecting it.

Step 2 Open the **Cloud History Export Manager** view, and click **All Descendants** so that all histories will be shown.

Step 3 Click **History Learn**.

This will add CloudHistoryExports for ALL the histories in the database.

Step 4 Select a single Cloud History Export and click **Archive**.

This will archive the single Cloud History Export's records to Sentience.

Step 5 If the previous step is successful, continue with each additional export, **one at a time**.

NOTE: If you know that the amount of data is not large, you may **conservatively** increase the number of history exports you select to expedite the process.

Step 6 Once completed for all cloud history exports, set the **`Batch Trigger Time`** property back to its original value. (The default is Interval - 15 minutes.).

WARNING: If you push too much data at one time, by selecting too many histories, Sentience **will** block the station from sending further data. In addition, you can cause blockages for other completely unrelated systems if too much of the Sentience IoTHub processing bandwidth is consumed by sending too many histories. Use great care when grouping history exports for archival.

Preparing alarms to send to the cloud

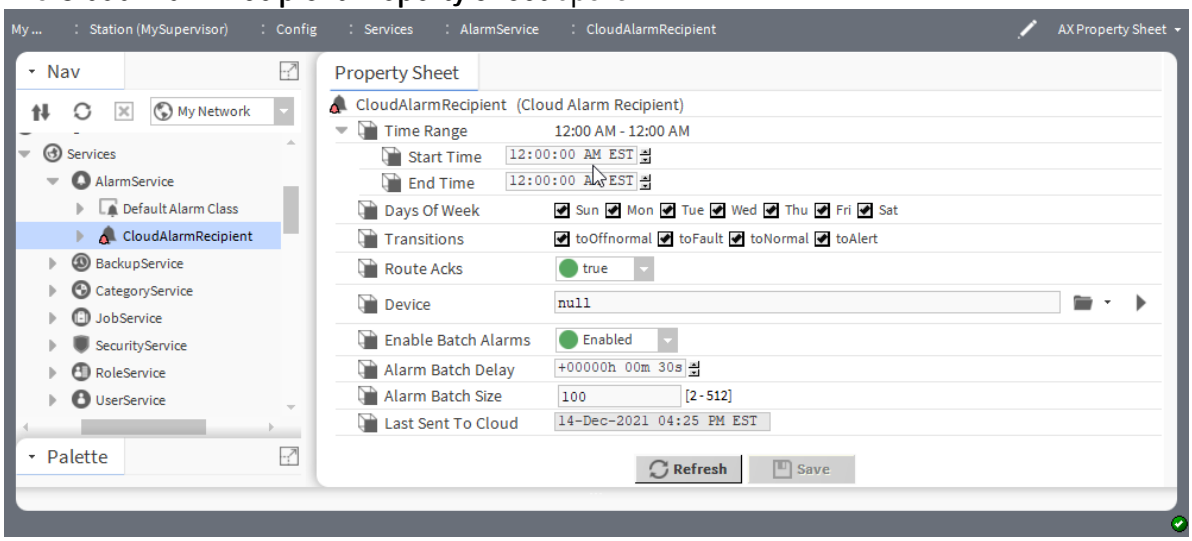
The **CloudAlarmRecipient** receives alarms from the framework as they are generated and sends them to the cloud using the configured cloud connector. The **`CloudSentienceDevice protocolType`** determines the protocol to use.

Prerequisites: The **`nCloudDriver`** palette is open.

Step 1 Expand the **Config→Services→AlarmService** and drag a **CloudAlarmRecipient** component from the palette to the **AlarmService**.

Step 2 Double-click the newly-added **CloudAlarmRecipient**

The **Cloud Alarm Recipient Property Sheet** opens.



Step 3 To connect the recipient with the cloud device, use the Ord Chooser to navigate to the **CloudSentienceDevice** in your **NiagaraCloudNetwork** and select it.

This **ORD** defaults to `null`, which prevents the cloud connector from routing alarms received by the recipient to Honeywell Sentience.

Step 4 If **Enable Batch Alarms** is disabled, enable it (the default), configure the **Alarm Batch Delay** and **Alarm Batch Size** properties and click **Save**.

As with all aspects of cloud communication, message bandwidth is a primary area factor in system performance. Sending a group of items is always preferred over sending a single item (or sending multiple messages, each containing a single item). Grouping items into a single batch request improves both the performance (due to more efficient network usage), and the cost (incurring fewer message sends).

The **CloudAlarmRecipient** groups alarms into batches when **Enable Batch Alarms** is enabled. When an alarm occurs, the station holds it for a brief time, configured by the **Alarm Batch Delay** and **Alarm Batch Size** properties. As soon as the recipient either receives a number of alarms greater than or equal to the batch size or as soon as the batch delay transpires, it sends the group of alarms to the cloud. You can place a maximum upper bound on the additional delay introduced. By default the batch delay is 30 seconds; this can be decreased if you require faster alarm delivery.

Performance optimization

Folder structure, batch size and update frequency impact throughput from the station to the cloud. You can adjust these factors for each individual station to find the optimum setting that maximizes the usage of each sent message while minimizing the amount of time required for the data transmission.

Folder structure may already be established in your stations. But, if not, you should consider how folder structure impacts performance. Stations with a flat folder structure have shorter point IDs, which means that more of them can fit in an update message. The trade-off is that a flat folder structure means more points are in each folder. Loading and viewing these folders might take longer. Stations with a deeper folder structure (or particularly with long folder names) fit fewer points in the update message (compression will mitigate this), which may load faster when viewing a particular folder.

The cloud point device extension (double-click the **Points** folder) defines the point ID based on a point's "handle." The property in question is **Use Handle For Point Id**.

Batch size (also defined on the cloud point device extension) specifies how many points to include in each update. It defaults to 500. Particularly for larger stations, you can increase this number without impacting performance. Settings beyond around 5000 may not produce significant improvements due to the Sentience

IoTHub's restrictions on maximum message size. This value depends on the station's structure, as the update includes the point ID, which depends upon the station's folder structure.

You can also configure the update interval for each of the three update groups.

Chapter 4 From the cloud to station

Topics covered in this chapter

♦ Requirements
♦ CloudWriteController and CloudWriteInfo
♦ Version compatibility matrix

Information moving from the cloud to the station is usually limited to commands that change individual points or configure individual properties. Large quantities of data rarely move from the cloud to a station.

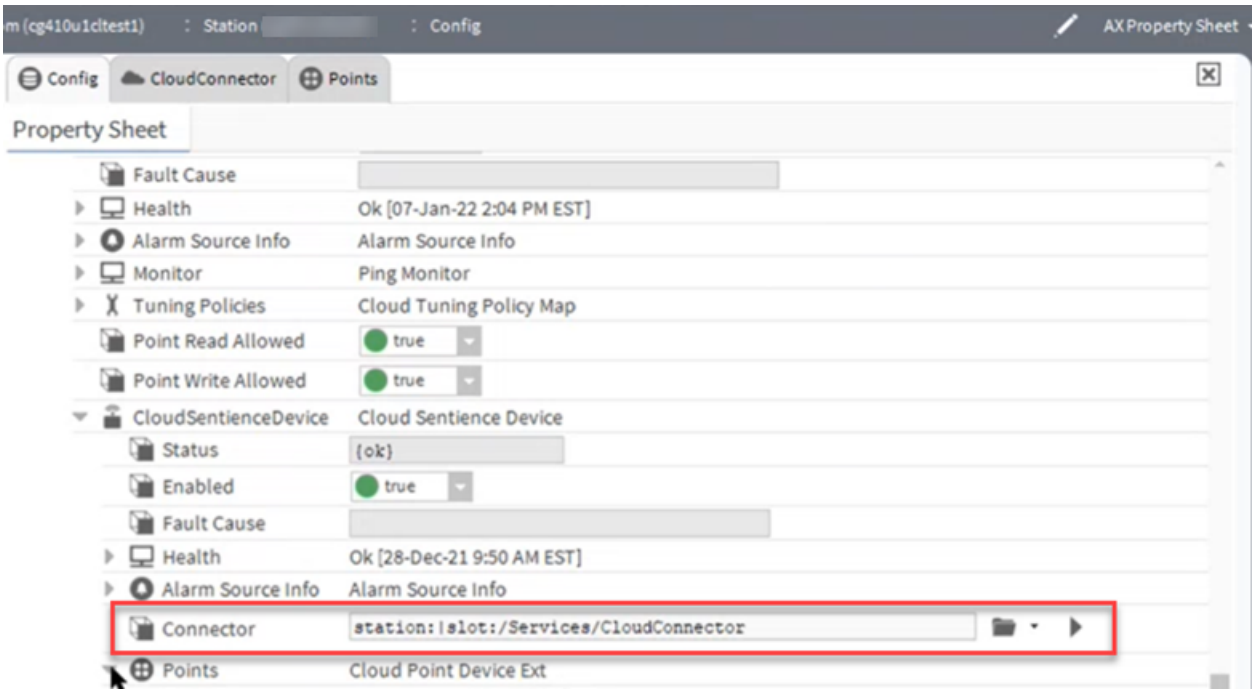
The following describes communications that are sent from the cloud to a station, including descriptions of how you can enable the **`cloudWriteController`** so that cloud applications can write values to remote stations from Forge Connect applications.

Requirements

The following section describes what you need to set up for using CloudWrite.

- The Forge Connect Gateway has been installed and registered with the cloud.
- In the Forge Connect Gateway, you have added the connector from the **`cloudSentienceConnector`** palette.
- You have a cloud network, for example, a NiagaraCloudNetwork, which contains the CloudSentienceDevice.
- Once you set up the CloudSentienceDevice, you need to point it at a cloud connector. More specifically, you have an ORD reference to the cloud connector through which the data are sent to the cloud.

  NOTE: In case of a single cloud connector, the device automatically locates and selects it.



- Prior to using CloudWrite, the Forge Connect device has discovered the remote station and its proxy points. The remote station has exported its points to the Forge Connect application using NCHSD

(Niagara Cloud Honeywell Sentience Driver), which means that the points are now replicated in the station's proxy points in the Forge Connect Gateway.

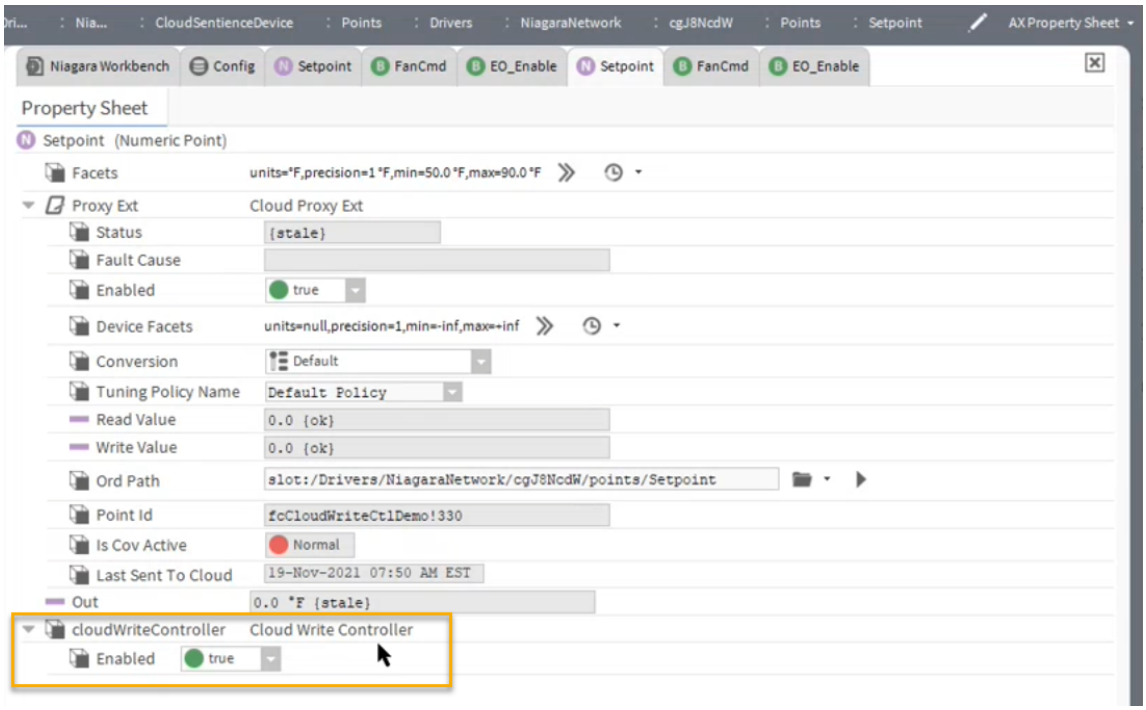
For more information about Easy Onboarding Service and model sync, see chapter "Cloud Connector registration and model sync" in the Honeywell Forge Connect Appliance Gateway Onboarding Guide.

CloudWriteController and CloudWriteInfo

This section describes the functions of the CloudWriteController and CloudWriteInfo.

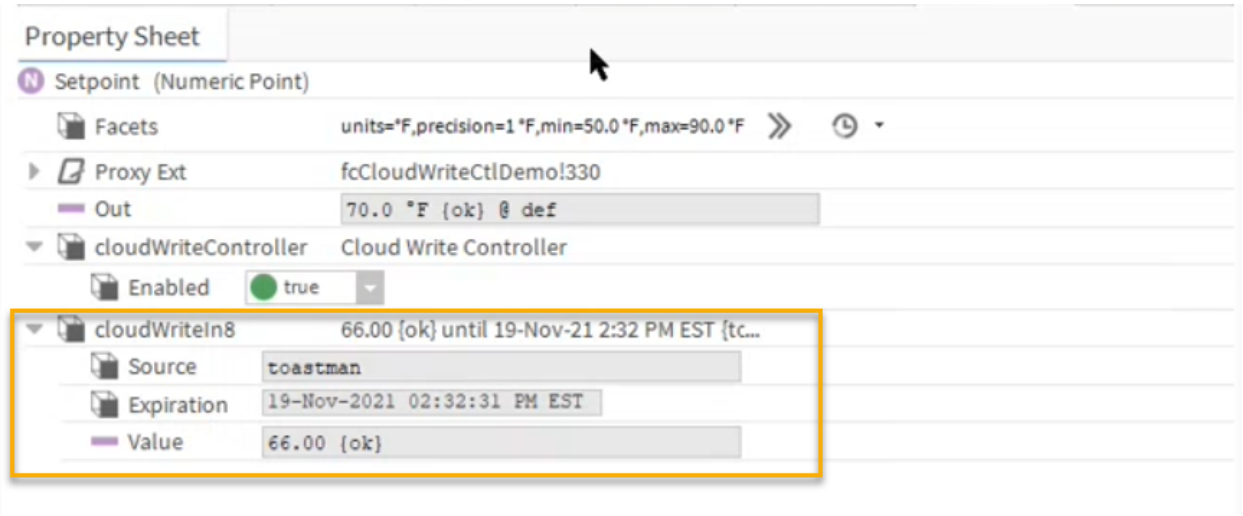
After you discovered and added a point to the CloudSentienceDevice in your cloud network, the cloud proxy extension automatically adds a **`cloudWriteController`**, which is enabled by default to allow write action.

Figure 1 CloudWriteController appearing when point added to CloudSentienceDevice



For each update (if the priority levels are unchanged) or write action that the cloud point receives from a Forge Connect application, the CloudWriteInfo component is added.

Figure 2 CloudWriteInfo generated upon CloudWrite action



The CloudWriteInfo represents all the information related to a particular CloudWrite instance as a child underneath the cloud proxy point.

The CloudWriteController only controls write action from the cloud through NCHSD (Niagara Cloud Honeywell Sentience Driver). It has a pluggable writer that is specific to the proxy point type, for example, a BacnetCloudWriter that is specific for a BACnet point or a FoxCloudWriter that writes to a Niagara point.

When you write data from the cloud, the Forge Connect Gateway writes them directly to the device. The integrated and pluggable writer of the enabled **`cloudWriteController`** accomplishes the write action by using, for example, the Fox channel. The **`cloudWriteInfo`** displays the value written by the cloud unless the local device overwrites the cloud value at the same priority.

Version compatibility matrix

This compatibility matrix provides information about the Forge Connect solution to connect to the Forge platform. The table shows the current Cloud Connector module and release versions and their compatibility with and inclusion in Niagara and Forge Connect. When you work with a particular Niagara version, make sure that you use the corresponding version of cloud modules.

Niagara version	Niagara modules version	Forge Connect release	NCHSD molules	Latest versions
4.10	4.10.0.X	1.3	2020.1.X	**NOTE:** 4.10.0 is EOL (end of life). Use 4.10u2 modules instead. cloudconnector: 2020.1.44 nclouddriver: 2020.1.36
4.10u1	4.10.1.X	1.3	2020.2.1.X	cloudconnector: 2020.2.1.12 nclouddriver: 2020.2.1.20
4.10u2	4.10.2.X	1.3	2020.2.2.X	cloudconnector: 2020.2.2.14 nclouddriver: 2022.2.36

4.11.0	4.11.0.X	n/a	2021.3.0.X	cloudconnector: 2021.3.0.10 nclouddriver: 2021.3.0.16
4.11u1	4.11.1.X	n/a	2021.11.1.X	cloudconnector: 2021.11.1.14 nclouddriver: 2021.11.1.18
4.12.0	4.12.0.X	n/a	2022.1.0.X (old) 2022.12.0.X	cloudconnector: 2022.12.0.16-RC nclouddriver: 2022.12.0.20-RC
4.13.0	4.13.0.X	n/a	2022.13.0.X	cloudconnector: 2022.13.0.10-RC nclouddriver: 2022.13.0.8-RC

Chapter 5 Security recommendations

Topics covered in this chapter

♦ Cloud-side system layers
♦ Station-side system layers

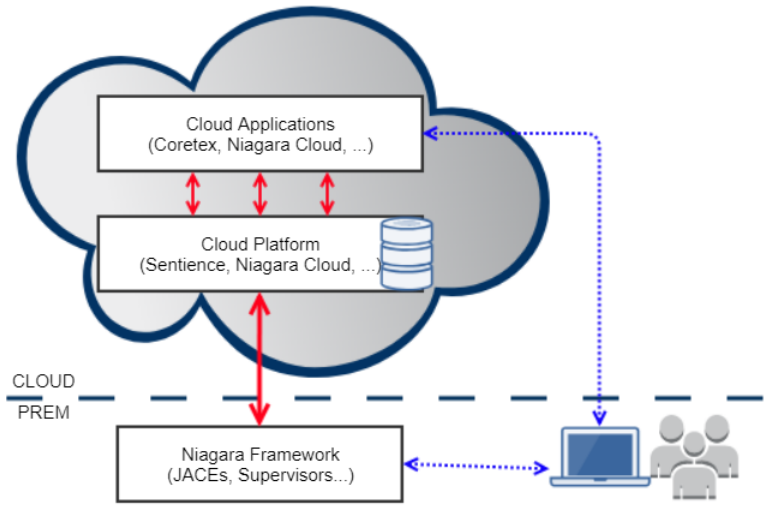
This chapter provides design recommendations and security requirements for developing custom cloud-to-device commands.

One of the goals for Niagara Cloud development is for the station to execute the commands received from the cloud (cloud application and/or cloud platform). This functionality opens multiple opportunities for developers to control and manage the station remotely from a cloud application, but it also creates challenges (station security, stability, etc.). This is especially important for the stations used as on-premise building command-and-control devices.

This chapter describes the chosen approach and steps required for stations to implement cloud commands.

One important point is that to secure the delivery and execution of cloud commends, actions are required on three system layers: in the cloud application, on the cloud platform, and in the **NiagaraStation**.

Figure 3 Message flow between on-premise device and cloud platform



Cloud applications provide the UI and serve various customer needs using cloud platform services.

Cloud applications send the commands on behalf of the user and receive a response from the station through the cloud platform. This means that cloud applications are in charge of user authentication and authorization (with or without cloud platform assistance). All other layers (cloud platform and station) improve the security but rely on the user information provided by the cloud application.

The cloud platform provides station registration (for Sentience 1.0 a System Guid is assigned for any registered station), cloud application authentication & authorization, and secure messages delivery to and from station.

Cloud-side system layers

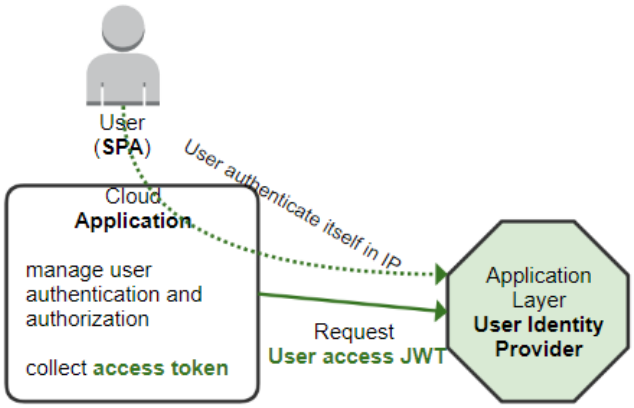
User authentication/authorization (application layer)

Cloud applications use different Identity Providers (IP) to authenticate the user. The cloud platform or any other trusted identity provider (enterprise or public) can provide an IP.

The application development team is fully responsible to choose the specific IP. The selected provider should support some standard OAuth functionality.

The cloud application should authorize the user unless the cloud platform provides the authorization. Most commonly, the cloud application should control user authorization for specific actions (sending a specific command to a particular station). Each user should have a defined role as a result of the authorization process.

Figure 4 User authentication — Application layer



As a result the cloud application should be able to collect a JSON Web Token (JWT) signed by the IP with these minimum claims:

- The algorithm should be RS256.
- The token issuer claim (iss) must contain a unique string that identifies the IP (typically this claim is fixed by the Identity Provider).
- The token audience claim (aud) must contain the predefined audience “Cloud Driver” (audience is configurable on the station side).
- The token subject claim (sub) should contain the user id (Use the guid for this. No personally identifiable information should be added to this field!).
- The private claim named “cloudroles” should contain the user role.
- The token expiration (exp) should be short! It is important to use short-living tokens to mitigate the risk of replay attacks. The recommendation is to choose a token time to live not longer then the typical user session.
- Avoid adding personally identifiable information to the JWT claims.

For development purposes, you could use native certificate management and `jwt.io`.

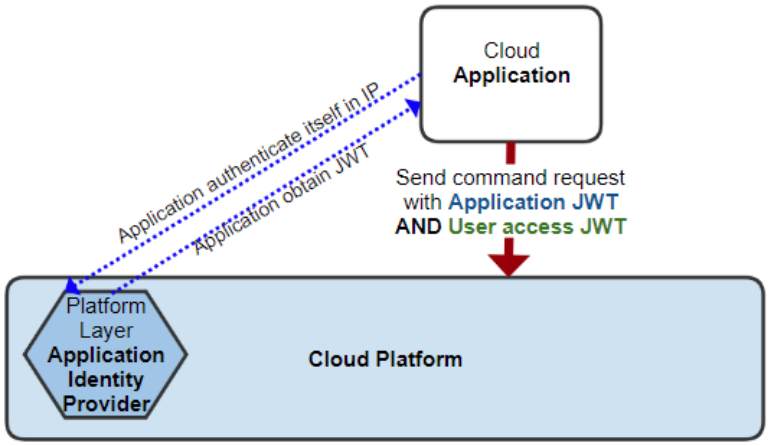
ACTION: Set up Identity Provider
Choose and configure the Identity Provider. Implement user authentication/authorization. The cloud application should be able to access JWT with particular claims to identify the user who initiated the command.

Application authentication/authorization (application and cloud platform layers)

The cloud application should be registered with the cloud platform to be able to use the platform API .

An authenticated cloud application can only send commands to a set of stations for which it (the application) has permission to send to each station in the set.

Figure 5 Application authorization with cloud platform



For Sentience1.0, an application can send commands one-by-one to specific **System GUIDs** with assigned permissions or to all devices of specific **System Type**. A Sentience DevOps request should be raised to register applications and provide **System Type** level permissions. **System GUID** permissions can be assigned by user with a particular role. Permission is required to send this command: **SystemCommandSend-SendSystemCommand**. There is a process to manually verify your application credentials.

ACTION: Register your Cloud application
Register your cloud application with cloud platform (Sentience) and assign required permissions for some set of stations (some **System Type** or list of **System GUIDs**). The cloud application should be able to obtain application token from cloud platform IP and use this token when accessing cloud platform APIs.

Send command (Application + Cloud Platform layers)

Receive command response (Application + Cloud Platform layers)

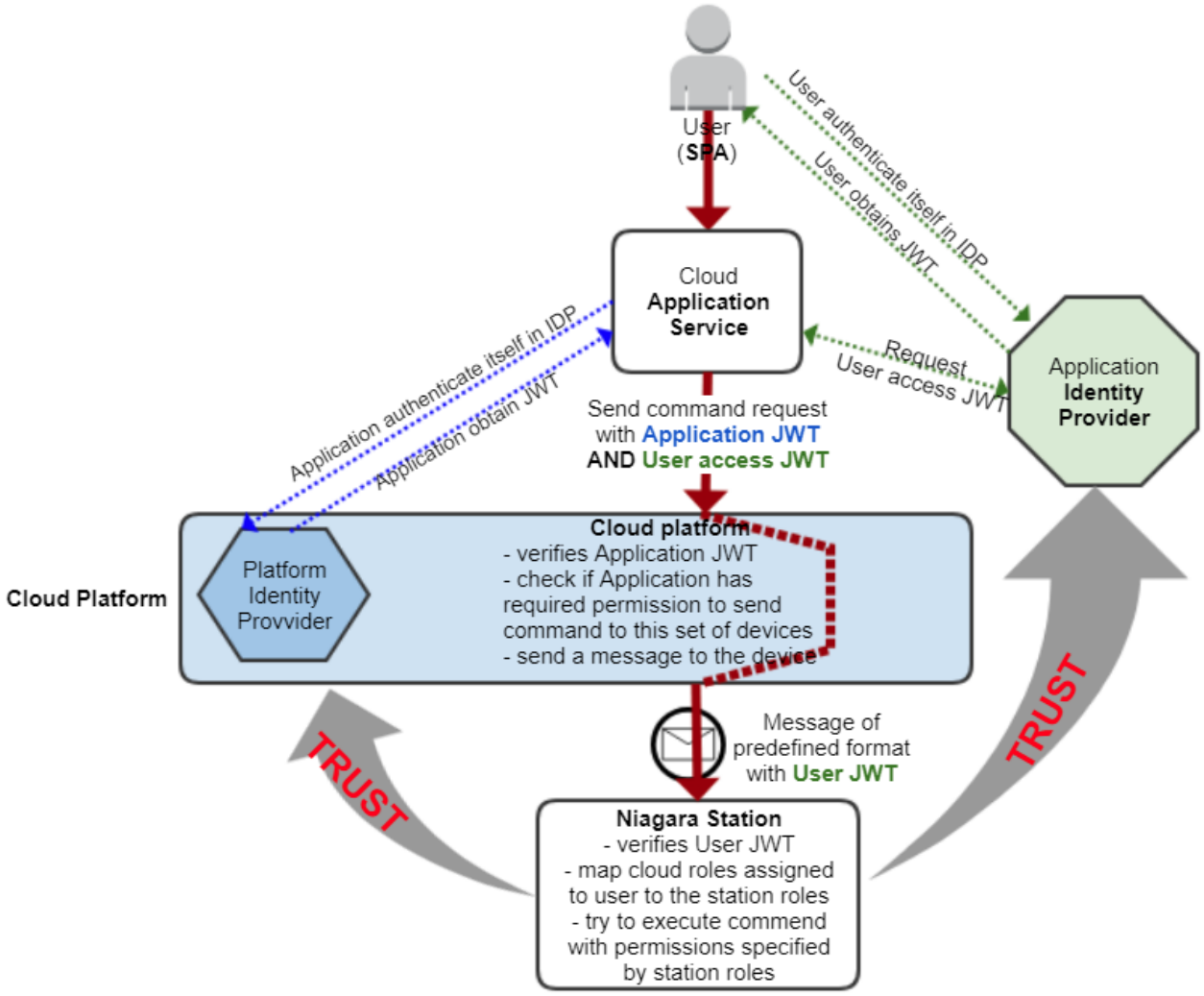
Station-side system layers

Each **NiagaraStation** registered with the cloud platform has a secured channel to send and receive messages using AMQPS/AMQPWS protocols.

Each **NiagaraStation** has its own authorization subsystem. To provide the required level of security, each command received by the station should be verified and logged by the customer-controlled, station-based authorization system.

This diagram shows the general flow for the cloud-to-device security commands.

Figure 6 Security commands general flow



The execution of cloud commands introduces additional security risk for the station. Famous use case: a cloud command, which writes boiler target temperature to 500 degrees.

Execution of all commands is disabled by default. To enable command execution, a customer or integrator must explicitly enable it.

ACTION

- `Enable Cloud Commands execution` configures NCD to enable execution of incoming cloud commands

Identity provider trust (station layer)

The device registration process configures trust in the platform identity provider. No additional steps are required.

Trust in the application layer user identity provider should be configured as a part of cloud commands configuration. This list summarize what needs to be done to configure this trust using the **Trust Manager**:

1. Provide the IP certificate or JWKS endpoint.
2. Fix the token issuer (iss) claim.

3. Possibly update the audience.
4. Collect and provide a certificate or JWKS URL for the chosen identity provider.

   Provide a token issuer and application IP and change the audience value, if needed.

   You configure trust for each cloud application. You should provide the application ID received from Sentience during the application registration process.

ACTION: Configure Identity Provider Trust
Collect and provide the certificate or JWKS URL for the chosen application IP. Provide the platform ID, which is equal to the Token issuer claim field. This should be unique for each application IP.

Cloud/station role mapping (station and application layers)

User roles for a cloud application (cloud roles) can be quite different from the security roles defined on the station (station roles).

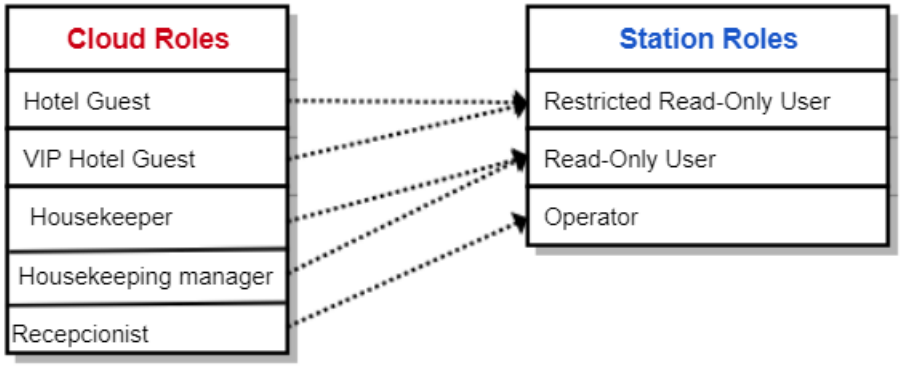
Cloud roles typically reflect user segmentation from the cloud application perspective and control user access (RBAC). Cloud roles are defined by cloud application developers and/or operations.

Station roles reflect segmentation from the particular station (building) perspective. The station security system uses a role-based access control approach to control user permissions. The station owner and/or integrator who installs and configures the station defines station roles.

To have the ability to control the execution of cloud commands, the station should know at least the user role and the mapping between user cloud role (provided as one of JWT claim) and some predefined station role (used to control permission on the station level).

For example, from the cloud application side, standard and VIP hotel guests could have different roles but, to the station they have the same role: extremely restricted user.

Figure 7 Cloud-to-station roles mapping



Important comments

1. The system rejects a command received without a cloud role or with a cloud role that is not mapped to a station role (refer to the overall station command flow diagram).
2. The system tries to execute, but rejects commands with a user context defined by a mapped station role whose station role permissions are not enough to perform the required action.
3. MVO3 has a different (incorrect) implementation. Instead of mapping, it directly uses the values of a JWT claim as station roles. MVO4 will correct this.

ACTION: Configure role mapping
Create reasonable station roles to provide the required level of security and access control. Configure the mapping of cloud roles to station roles.

Overall station command flow (station layer)

Shown here is the token verification and command execution flow.

Figure 8 Incoming command message flow

System Disabled

The Honeywell Forge Operations team tracks and aggressively manages the bandwidth usage of systems participating in the Forge Platform ecosystem. Devices that send too much data are subject to being disabled from the Forge IoTHub. This means that the device is prevented from sending any data to the Forge IoTHub. The device may even be prevented from establishing the IoTHub connection in the first place. This may manifest in several different ways. One example is where the authenticator is able to authenticate to the identity endpoint, but cannot open the connection. You may see the Property Sheet of the RpkAuthenticator show "Connected", or possible "Pending Connect". The following message, or something similar, may show in the station output:

```
CONFIG [14:33:56 29-Jul-19 BST][cloud.connector.sentience] Starting RPK Challenge
CONFIG [14:33:56 29-Jul-19 BST][cloud.connector.sentience] Sending RPK Challenge request to URI https://gaprodsystemauthentication.s
FINE [14:33:57 29-Jul-19 BST][cloud.connector.http] HTTP Response Code:200
CONFIG [14:33:57 29-Jul-19 BST][cloud.connector.sentience] Checking for existing locally initialized keys
CONFIG [14:33:58 29-Jul-19 BST][cloud.connector.sentience] Authenticating using software keys
CONFIG [14:33:58 29-Jul-19 BST][cloud.connector.sentience] Sending RPK Challenge Response to URI https://gaprodsystemauthentication.
FINE [14:33:59 29-Jul-19 BST][cloud.connector.http] HTTP Response Code:200
CONFIG [14:33:59 29-Jul-19 BST][cloud.connector.sentience] Completed RPK Challenge - 2438 ms
CONFIG [14:33:59 29-Jul-19 BST][cloud.connector.sentience] Starting System Connections
WARNING [14:33:59 29-Jul-19 BST][com.microsoft.azure.sdk.iot.device.transport.amqps.AmqpsDeviceAuthenticationCBSTokenRenewalTask] jav
FINE [14:34:02 29-Jul-19 BST][cloud.connector.http] HTTP Response Code:200
CONFIG [14:34:02 29-Jul-19 BST][cloud.connector.sentience] Completed System Connections: 3125 ms

FINEST [14:34:03 29-Jul-19 BST][cloud.connector] BCloudConnector.pingFail(Could not open the connection), notifying connectCallbacks
FINE [14:34:03 29-Jul-19 BST][cloud.connector] Connection fail
java.io.IOException: Could not open the connection
    at com.microsoft.azure.sdk.iot.device.DeviceIO.open(DeviceIO.java:165)
    at com.microsoft.azure.sdk.iot.device.DeviceClient.open(DeviceClient.java:369)
    at com.tridium.cloud.client.iotdep.BIotHubMessageClient.lambda$onConnect$4(BIotHubMessageClient.java:441)
    at java.security.AccessController.doPrivileged(Native Method)
    at com.tridium.cloud.client.iotdep.BIotHubMessageClient.onConnect(BIotHubMessageClient.java:387)
    at com.tridium.cloud.client.iothub.BAbstractIotHubConnectorImpl.doConnect(BAbstractIotHubConnectorImpl.java:79)
    at com.tridium.cloud.client.sentience.BSentienceConnectorImpl.doConnect(BSentienceConnectorImpl.java:734)
    at com.tridium.cloud.client.BConnectorImpl.connect(BConnectorImpl.java:118)
    at com.tridium.cloud.client.BCloudConnector.reconnectSync(BCloudConnector.java:527)
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)
    at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$201(ScheduledThreadPoolExecutor.java:180)
    at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:293)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at java.lang.Thread.run(Thread.java:748)
Caused by: com.microsoft.azure.sdk.iot.device.exceptions.TransportException: Unknown transport exception occurred
    at com.microsoft.azure.sdk.iot.device.transport.amqps.AmqpsIotHubConnection.onLinkRemoteClose(AmqpsIotHubConnection.java:729)
    at org.apache.qpid.proton.engine.BaseHandler.handle(BaseHandler.java:176)
    at org.apache.qpid.proton.engine.impl.EventImpl.dispatch(EventImpl.java:108)
    at org.apache.qpid.proton.reactor.impl.ReactorImpl.dispatch(ReactorImpl.java:324)
    at org.apache.qpid.proton.reactor.impl.ReactorImpl.process(ReactorImpl.java:291)
    at com.microsoft.azure.sdk.iot.device.transport.amqps.IotHubReactor.run(IotHubReactor.java:28)
    at com.microsoft.azure.sdk.iot.device.transport.amqps.AmqpsIotHubConnection$ReactorRunner.call(AmqpsIotHubConnection.java:824)
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)
    ... 3 more
```

Important notes

1. Different priority queues execute important commands first. The message payload defines priority.
2. For command responses with multiple messages a subscription to the annotation stream is required.

WARNING: The workflow shown in the dashed line in the diagram is not secure and is not recommended by Tridium. This approach is not approved by Honeywell security architects!

Predefined commands (station and application layers)

Some commands are already implemented as a part of standard **`nCloudDriver`** functionality and can be invoked without additional coding. These commands are developed by the Tridium development team and have passed the required security tests.

For Sentience 1.0 these are system commands, each with a particular ID.

Command Id	Command description	Command parameters	Command Response
Replicas of Sentience 1.0 "native" commands to be used with "secured execution"			
AlarmAckCommand	Acknowledge the Alarm	COMMAND_PAYLOAD.ALARMID **CommandPayload.AlarmId**	Returns Status and Message for the Alarm Ack
CloudPointReadCommand	Read value of one point	COMMAND_PAYLOAD.POINTID **CommandPayload.PointId**	Returns value, quality and timestamp for one point
CloudPointReadInputsCommand	Reads the active inputs to a cloud proxy point.	CommandPayload.PointId	Returns Source, Priority, Value, and Expiration for each active cloud write to the point.
CloudMultiPointReadInputsCommand	Read active writes to a group of points	CommandPayload.PointIds	Returns Source, Priority, Value, and Expiration for each active cloud write to each point.
CloudMultiPointReadCommand	Read values of multiple points	COMMAND_PAYLOAD.POINTIDS CommandPayload.PointIds	Returns values, quality and timestamp for multiple points
CloudPointWriteCommand	Write value of one point	COMMAND_PAYLOAD.POINTID, COMMAND_PAYLOAD.VALUE **CommandPayload.PointId CommandPayload.Value CommandPayload.Priority** **CommandPayload.Duration** **CommandPayload.Source**	Returns result of the Write operation (Success/Fail) for one point
CloudMultiPointWriteCommand	Write values of multiple points	COMMAND_PAYLOAD.POINTWRITES{POINTID, Value} **CommandPayload.PointWrites {PointId, Value, Priority, Duration, Source}**	Returns results of the Write operation (Success/accessNotAllowed) for multiple points
Additional predefined commands			
BatchAlarmAckCommand	Acknowledge multiple Alarms (identified by Alarm Ids)	COMMAND_PAYLOAD.ALARMS CommandPayload.Alarms	Returns Status and Message for multiple Alarm Acks
CloudMultiPointClearCommand	Clear points value to null	COMMAND_PAYLOAD.POINTIDS CommandPayload.PointIds	Returns results of the Clear operation (Success/accessNotAllowed) for multiple points

Command Id	Command description	Command parameters	Command Response
CovActiveCommand	Change Point to COV mode	COMMAND_PAYLOAD.POINTIDS CommandPayload.PointIds	Returns the result of changing point mode operation for each point
CovInactiveCommand	Change Point back from COV to snapshot mode	COMMAND_PAYLOAD.POINTIDS CommandPayload.PointIds	Returns the result of changing point mode operation for each point
InvokeCommand	Execute Action on Component	TO BE UPDATED LATER	TO BE UPDATED LATER
RetrieveCloudPointsCommand	Retrieve the list of all cloud exposed points	-	Returns list of Ids for all cloud exposed points
SubscribePointsCommand	Place one or more cloud points' associated station points into subscription	**CommandPayload.PointIds**	Returns list of successes and failures for the point subscribes.
UnsubscribePointsCommand	Place one or more cloud points' associated station points out of subscription	**CommandPayload.PointIds**	Returns list of successes and failures for the point unsubscribes.

ACTION: Use predefined command ids and parameters
Issue commands with predefined IDs and correct parameters using the standard cloud platform APIs (System Command API for Sentience 1.0). Listen for command response events (Annotation Stream for Sentience 1.0)

Predefined command responses (station and application layers)

All system commands respond with Sentience events. The fields contained in the event are common among commands and are similar to the following sample response.

Figure 9 Predefined command response

```
{
  "CloudPlatformEvent":{
    "CreatedTime":"<<Timestamp>>",
    "Id":"<<Event guid>",
    "CreatorId":"NiagaraFramework",
    "CreatorType":"Niagara4",
    "GeneratorId":"<<System guid>>",
    "GeneratorType":"NiagaraCloudConnector",
    "TargetId":"<<Sentience MessageId>>",
    "TargetType":"Device",
    "TargetContext":"System Command",
    "Body":{
      "type":"TextualBody",
      "value":"<<JSON encoded string with command specific response>>",
      "format":"application/json"
    },
    "BodyProperties":[
      {"Key":"SystemType","Value":"<<Station Type>>"},
      {"Key":"SystemGuid","Value":"<<System guid>>"}
    ],
    "EventType":"system command update"
  },
  "AnnotationStreamIds":","
}
```

Once the JSON in the body's value field has been converted from a string back into JSON, encoding it appears as follows for each predefined command.

Command Id	Command Response

AlarmAckCommand	{
	"AlarmId":"<<Id of the alarm that was acknowledged>>",
	"CommandId":"<<Command Id sent in the cloudPlatformHeadders>>",
	"CorrelationId":"<<Sentience Message Id>>",
	"ErrorMessage":"<<Error message or 'Status: OK'>>",
	"IsSuccessful":<<true or false>>,
	"Message":"<<Alarm acknowledgement status message>>",
	"ResponseCode":200,
	"Status":"<<'Success' or 'Fail'>>"
	}

BatchAlarmAckCommand	{
	"Alarms":{
	"<<Alarm Id 1>>":{
	"IsSuccessful":<<true or false>>,
	"Message":"<<Alarm acknowledgement status message>>"
	},
	"<<Alarm Id 2>>":{ ... },
	...,
	"<<Alarm Id N>>":{ ... }
	},
	"CommandId":"<<Command Id sent in the cloudPlatformHeadders>>",
	"CorrelationId":"<<Sentience Message Id>>",
	"ErrorMessage":"<<Error message or 'Status: OK'>>",
	"ResponseCode":200,
	"Status":"<<'Success' or 'Fail'>>"
	}

CloudMultiPointClearCommand	{
	"CommandId":"<<Command Id sent in the cloudPlatformHeadders>>",
	"CorrelationId":"<<Sentience Message Id>>",
	"ErrorMessage":"<<Error message or 'Status: OK'>>",
	"PointClearDetails":{
	"<<Point Id 1>>":"<<'Success' or 'Fail'>>",
	"<<Point Id 2>>":"<<'Success' or 'Fail'>>",
	...,
	"<<Point Id N>>":"<<'Success' or 'Fail'>>"
	},
	"ResponseCode":200,
	"segmentTotal":<<Total number of 200 response code messages to expect for this command>>,
	"SequenceNum":<<The number of this response in the range 0 to segmentTotal - 1>>
	}

CloudMultiPointReadCommand	{
	"CommandId":"<<Command Id sent in the cloudPlatformHeadders>>",
	"CorrelationId":"<<Sentience Message Id>>",
	"ErrorMessage":"<<Error message or 'Status: OK'>>",
	"PointValues":{
	"<<Point Id 1>>":{
	"Status":"<<Point status>>",
	"Value":<<Point value>>
	},
	"<<Point Id 2>>":{ ... },
	...,
	"<<Point Id N>>":{ ... }
	},
	"ResponseCode":200,
	"segmentTotal":<<Total number of 200 response code messages to expect for this command>>,
	"SequenceNum":<<The number of this response in the range 0 to segmentTotal - 1>>
	}

Command Id	Command Response

CloudMultiPointReadInputsCommand	{
	"CommandId":"<<Command Id sent in the cloudPlatformHeadders>>",
	"CorrelationId":"<<Sentience Message Id>>",
	"ErrorMessage":"<<Error message or 'Status: OK'>>",
	"<<Point Id>>":{
	"Source":"<<Point source >>",
	"Priority":"<<Point priority>>",
	"Value":"<<Point value>>",
	"Expiration":"<<Point expiration >>
	},
	"ResponseCode":200
	}

CloudMultiPointWriteCommand	{
	"CommandId":"<<Command Id sent in the cloudPlatformHeadders>>",
	"CorrelationId":"<<Sentience Message Id>>",
	"ErrorMessage":"<<Error message or 'Status: OK'>>",
	"PointWriteDetails":{
	"<<Point Id 1>>":"<<'Success' or 'Fail'>>",
	"<<Point Id 2>>":"<<'Success' or 'Fail'>>",
	...,
	"<<Point Id N>>":"<<'Success' or 'Fail'>>"
	},
	"ResponseCode":200,
	"segmentTotal":<<Total number of 200 response code messages to expect for this command>>,
	"SequenceNum":<<The number of this response in the range 0 to segmentTotal - 1>>
	}

CloudPointReadCommand	{
	"CommandId":"<<Command Id sent in the cloudPlatformHeadders>>",
	"CorrelationId":"<<Sentience Message Id>>",
	"ErrorMessage":"<<Error message or 'Status: OK'>>",
	"<<Point Id>>":{
	"Status":"<<Point status>>",
	"Value":<<Point value>>
	},
	"ResponseCode":200
	}

CloudPointReadInputsCommand	{
	"CommandId":"<<Command Id sent in the cloudPlatformHeadders>>",
	"CorrelationId":"<<Sentience Message Id>>",
	"ErrorMessage":"<<Error message or 'Status: OK'>>",
	"<<Point Id>>":{
	"Source":"<<Point source >>",
	"Priority":"<<Point priority>>",
	"Value":"<<Point value>>",
	"Expiration":"<<Point expiration >>
	},
	"ResponseCode":200
	}

Command Id	Command Response

CloudPointWriteCommand	{
	"CommandId":"<<Command Id sent in the cloudPlatformHeadders>>",
	"CorrelationId":"<<Sentience Message Id>>",
	"ErrorMessage":"<<Error message or 'Status: OK'>>",
	"<<Point Id>>":"<<'Success' or 'Fail'>>",
	"ResponseCode":200
	}

CovActiveCommand	{
	"CommandId":"<<Command Id sent in the cloudPlatformHeadders>>",
	"CorrelationId":"<<Sentience Message Id>>",
	"ErrorMessage":"<<Error message or 'Status: OK'>>",
	"PointCOVStatuses":{
	"<<Point Id 1>>":<<true or false>>,
	"<<Point Id 2>>":<<true or false>>,
	...,
	"<<Point Id N>>":<<true or false>>
	},
	"ResponseCode":200
	}

CovInactiveCommand	{
	"CommandId":"<<Command Id sent in the cloudPlatformHeadders>>",
	"CorrelationId":"<<Sentience Message Id>>",
	"ErrorMessage":"<<Error message or 'Status: OK'>>",
	"PointCOVStatuses":{
	"<<Point Id 1>>":<<true or false>>,
	"<<Point Id 2>>":<<true or false>>,
	...,
	"<<Point Id N>>":<<true or false>>
	},
	"ResponseCode":200
	}

InvokeCommand	

RetrieveCloudPointsCommand	{
	"CommandId":"<<Command Id sent in the cloudPlatformHeadders>>",
	"CorrelationId":"<<Sentience Message Id>>",
	"ErrorMessage":"<<Error message or 'Status: OK'>>",
	"points":["<<Point Id 1>>", "Point Id 2", ..., "Point Id N"]
	"ResponseCode":200,
	"segmentTotal":<<Total number of 200 response code messages to expect for this command>>,
	"SequenceNum":<<The number of this response in the range 0 to segmentTotal - 1>>,
	"Status":"<<Status message>>"
	}

RetrieveCloudCommandsCommand	

Command Id	Command Response

SubscribeCommand	{
	"CommandId":"<<Command Id sent in the cloudPlatformHeadders>>",
	"CorrelationId":"<<Sentience Message Id>>",
	"ErrorMessage":"<<Error message or 'Status: OK'>>",
	"Point Id":{
	"<<Point Id 1>>":<<true or false>>,
	"<<Point Id 2>>":<<true or false>>,
	...,
	"<<Point Id N>>":<<true or false>>
	},
	"ResponseCode":200
	}

UnsubscribeCommand	{
	"CommandId":"<<Command Id sent in the cloudPlatformHeadders>>",
	"CorrelationId":"<<Sentience Message Id>>",
	"ErrorMessage":"<<Error message or 'Status: OK'>>",
	"Point Id":{
	"<<Point Id 1>>":<<true or false>>,
	"<<Point Id 2>>":<<true or false>>,
	...,
	"<<Point Id N>>":<<true or false>>
	},
	"ResponseCode":200
	}

Develop your own custom command

The ability to invoke custom code provides the greatest flexibility in exchange for great responsibility. Keep in mind the multiple restrictions of the underlying hardware and security requirements.

IMPORTANT: The developer of a custom command is in charge of implementing the required security restrictions and of ensuring adherence to the stability requirements! Review the predefined commands before you start a custom command development!

To simplify development and to improve code quality, the BCloudCustomCommand allows Tridium to enforce certain security needs including auditing, permission checking, etc.

To develop your own custom command:

- Extend the BCloudCustomCommand class.
- Override the runCommand method to implement the required command behavior.
- Create a unique command name and register it.

Custom commands introduce an additional high security risk for the customer (customers do not really know what will be executed on their stations). To mitigate this risk, the system disables the execution of custom commands by default.

The BCloudCustomCommand provides a separate flag to enable and disable of this functionality.

Some important requirements

- Do not start additional threads in your custom command. A custom command is not expected to be multi-threaded!

- Catch InterruptedExceptions in your custom command code. This exception means that your command should stop execution and exit. There are multiple reasons why the framework can send this exception to your code, but you have to follow this rule.

ACTIONS: Develop your own command module
Create your own module with a custom command by extending the BCloudCustomCommand class. Install your module in a station. Invoke your own command using your own command ID.

Chapter 6 Maintenance

Topics covered in this chapter

- ♦ Workflow considerations
- ♦ Station copy
- ♦ Host upgrade
- ♦ Distributing NCHSD modules to users
- ♦ Confirming a module using 7-Zip

This chapter covers basic guidelines for upgrading the stations connected to the cloud.

Workflow considerations

Recommended best practices to minimize interruption in operations when copying stations, upgrading a host, and distributing NCHSD modules among users.

The cloud connector contains an ID that is unique to the station. This is known in Sentience terminology as the "System Guid" (Globally Unique IDentifier). Registration by the **Niagara Cloud Device Registration** web service assigns this ID to the system. The System Guid is tied to the **`System ID`** that can be found in the cloud connector's **`ConnectorImpl`** property as well as the **`Public/Private Key Pair`** that is stored securely in the **Niagara Certificate Manager User Key Store**. It is important to maintain all of these together. Removing one eliminates the ability of the connector to establish its identity with the Sentience identity provider, and thereby prevents the station from communicating with the cloud. The **`Alias`** used for the key in the **User Key Store** is the station's name, whereas the **`Subject`** is the **`System ID`** of the connector.

The **`System ID`** is established by combining the station name with the host ID of the platform, which generates a unique string to register with the Sentience identity provider.

Station copy

When copying stations from one host to another, you will need to register the cloud connector on the new host.

The connector updates its **`System ID`** to reflect the new host's host ID, and establishes a new **`Public/Private Key Pair`**. You must register this information with the Sentience identity provider.

CAUTION: When you re-register your station, its cloud connector receives a new, and completely independent **`System Guid`**. This means that any data from the old registration are not associated with the new connector id. This can be of concern when trying to connect data from before the re-registration with data from after the re-registration.

Host upgrade

Pitfalls to be aware of before upgrading a host.

NOTE: Before you upgrade a station, store the Sentience system's public and private keys in a secure location.

If you install a clean distribution file on a JACE during the upgrade, you will lose the key pair (the Sentience system's public and private keys) that enable the station to connect to the cloud even after installing the cloud connector with the same **`System ID`**.

When upgrading a host, it is not necessary to install a clean distribution file on JACE; this is only needed when you are downgrading a JACE to an earlier version of Niagara. If you do not install a clean distribution

file on the JACE, and you leave the station in place, it should retain the key pair and reconnect after the upgrade with no problem.

If you need to install a clean distribution file on the JACE, you will need to save the keys. To save the keys:

1. In the station, click **Services→PlatformServices→CertManagerService**
2. On the **User Key Store** tab, locate the key for the station. This will have an alias in a specific format (see the table below).
3. Select the key and click **Export** to export it.
   a. Be sure to click to select both checkbox options: "**Export the public certificate**" and "**Export the private key**".

      Also, you may want to select the option to "**Encrypt the exported private key**" and provide a password, particularly if this is a production system.

   b. Click **OK** and then **Save** the key as a `.pem` file somewhere accessible to you.

Alias name

Note the alias name. It is important that you install the CloudConnector key with the exactly correct alias name, because that is what the connector will look for. If it cannot find the name exactly as it expects, it will create a new key with the correct alias, and will not be able to connect as this key is not what is registered with the cloud identity provider.

NOTE: The alias name is used as the suggested file name for the `.pem` file. Note, that this may not be an allowed filename, so you may have to change the name. If so, make sure to note the alias name. For example, the alias generally will contain colon (`':'`) characters, which are not allowed for the filename. You will probably need to change these to underscores to save the filename. It is important to change the underscores (or whatever substitution character you use) back to colons when importing the key into the alias.

The correct CloudConnector Key Alias format depends on the installation version, as shown in the following table.

Niagara Version / Product	Alias Format	Subject	See Note
Niagara 4.4 (NCHSD 2018.5.*)	<station-name>	N4:**<station-Name>:<host-id>**	1, 2, 4
Niagara 4.6, Niagara 4.7 (NCHSD 2018.6.*)	<station-name>	N4:**<station-Name>:<host-id>**	1, 2, 4
Niagara 4.8 (NCHSD 2019.1) - JACE, Supervisor	cloud_n4:**<station-name>:<host-id>**	N4:**<station-Name>:<host-id>**	1, 2, 4
Niagara 4.8 (NCHSD 2019.1) - Gateway	cloud_guid:**<station-guid>**	GUID:**<station-guid>**	3
Niagara 4.9 (NCHSD 2019.2) - JACE, Supervisor	cloud_n4:**<station-name>:<host-id>**	N4:**<station-Name>:<host-id>**	1, 2, 4
Niagara 4.9 (NCHSD 2019.2) - Gateway	cloud_guid:**<station-guid>**	GUID:**<station-guid>**	3

- **<station-name>** = Station Name, with lowercase letters and underscores (_) replaced with hyphens (-)
- **<station-Name>** = Station Name, with original case, and underscores (_) replaced with hyphens (-)
- **<station-guid>** = Station unique GUID (lowercase hexadecimal characters and hyphens only), used as a unique identifier for platforms with shared host ids
- **<host-id>** = Platform Host ID

Once the upgrade is completed, reinstall the keys **BEFORE** starting the station:

1. With the station idle, open a Platform connection and click **Certificate Management→User Key Store** view.
2. Ensure there are no keys with the alias you need to use. Delete them if needed.

3. Click **Import**.
4. Select the `.pem` file you exported before upgrading and click **Open**. Enter the password if needed.
5. In the **Certificate Import** dialog, make sure to enter the station name alias EXACTLY as it is expected from the table above for the key, if you used a filename other than the suggested one (which matches the station name). Click **OK** when finished.
6. Start the station.

NOTE: **`Alias Name`** is important. Make sure that the alias name used for the key import is correct, according to the table above. You may need to recall the alias name saved from the key export, if the filename does not match the required alias name.

If you already cleaned your JACE

If you have already cleaned your JACE, then the keys for the CloudConnector are lost. You need to contact Technical Support to get your station de-registered, so you can re-register it with Niagara Cloud. Note that this will give you a new registration which is not connected to the old data. It may be possible to connect the old data to the new data with support from the Digital Operations team.

Distributing NCHSD modules to users

When distributing NCHSD modules, and indeed ANY Niagara module, to users, be aware of potential pitfalls that may arise due to interactions with the sender's and recipient's local IT administration.

The IT departments may often implement algorithms to protect users from potentially dangerous files that may be sent maliciously through email. Unfortunately, Niagara files are of the same type as some of the common forms of malicious files, for example `.jar`, `.js`, `.exe`, and `.zip` files. For this reason, it is important to ensure that the files are delivered intact to the recipient. There are several ways to do this. The main goal is to avoid the scanning of the email algorithms:

- One way that can increase the likelihood that the file will be delivered intact to the intended recipient is to zip the file, and apply an AES-256 encrypted password to the file. This prevents the file from being tampered with by the protection algorithms in most cases. Assuming you have 7-zip installed, you can do the following:

  1. Right-click on the file, or on the folder containing the group of files to be sent. Click **7-Zip→Add to archive**.
  2. In the **Encryption** section, enter a password, and ensure that the selected **`Encryption`** method is `AES-256`. It is not necessary to select Encrypt file names.
  3. Click **OK**.

     You can then email the zipped file which should be safe from having files stripped.

- Another way is to use **`OneDrive`**, or **`SharePoint`**, to distribute the module to the recipient. If the recipient is not part of the Honeywell corporate network, perhaps another file sharing option can be used.
- It would also be possible to use a USB flash drive to distribute the module, although this is only useful for a small number of recipients at a time, and also may require special USB Write permissions to write the module to the USB flash drive.

  NOTE: Remember to follow safe practices regarding the usage of USB drives. Do not use a drive whose origins you are unsure of; ensure that it has been scanned, or otherwise confirmed safe and virus-free before using!

Confirming Niagara modules after delivery

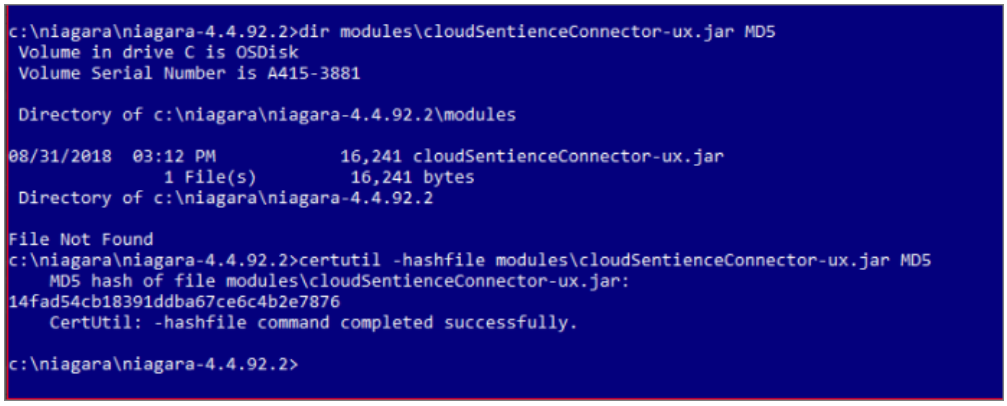
Unless you know what the contents of the module should be, it is difficult to confirm that the module is fully intact. There is no general rule to follow to check modules. However, signature verification can validate any classfiles that are loaded. There is one module in the NCHSD image that contains JavaScript files that are often the target of the protection algorithms' attention: `cloudSentienceConnector-ux.jar`. Confirm that the JavaScript files have not been stripped from this module before using.

To confirm that the module is valid, you can open a Niagara **Console** window, and enter two commands:

- `dir modules\cloudSentienceConnector-ux.jar`
- `certutil -hashfile modules\cloudSentienceConnector-ux.jar`

The output should look like this (note the exact size and hash values are version dependent; see the table below for the expected values).

Figure 10 Confirming Javascript files in module



If the hash or size of the file does not match the expected value, obtain an uncorrupted version of the file.

This table shows the expected file size and hash values for `cloudSentienceConnector-ux` module:

Module Version	File Size (B)	MD5 Hash
2018.4.2	16241	`14fad54cb18391ddba67ce6c4b2e7876`
2018.5.4	16624	`074b73f1962c9a0e1c93af5847c761d0`
2018.5.5	16615	`96a09664cb0a86046d9b1122d2b3bdc4`
2018.5.6	16614	`c8f03b1b9b2dc671e53eb99c3d97dc7d`
2018.6.6	16618	`dd3b3cb508edb9b22564181b6d1831f4`
2018.6.7	16616	`f7aa21b5969a835ffc15bb1d13ddbf73`

The "Module confirmation using 7–Zip" procedure (in this section) can be used to check on the contents of the file. Note that doing this is not necessary once you have confirmed a hash/size mismatch, since you know the file is corrupt.

If your module has been corrupted, you will not be able to use it to view the **`Device Registration Widget`**, or register a CloudConnector. You may be able to use the software image for other NCHSD activities, but it is recommended that you replace this with a valid module before doing any other activities with this Niagara installation.

Confirming a module using 7-Zip

You can confirm the files in the `cloudSentienceConnector-ux` module using 7-Zip.

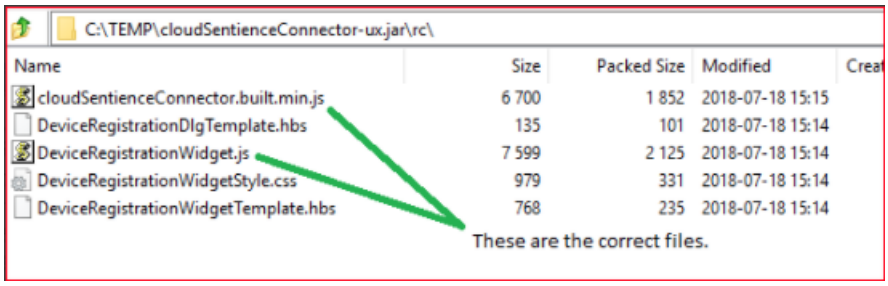
Prerequisites:

CAUTION: Take care not to modify any of the contents of the file, or you may prevent it from working correctly.

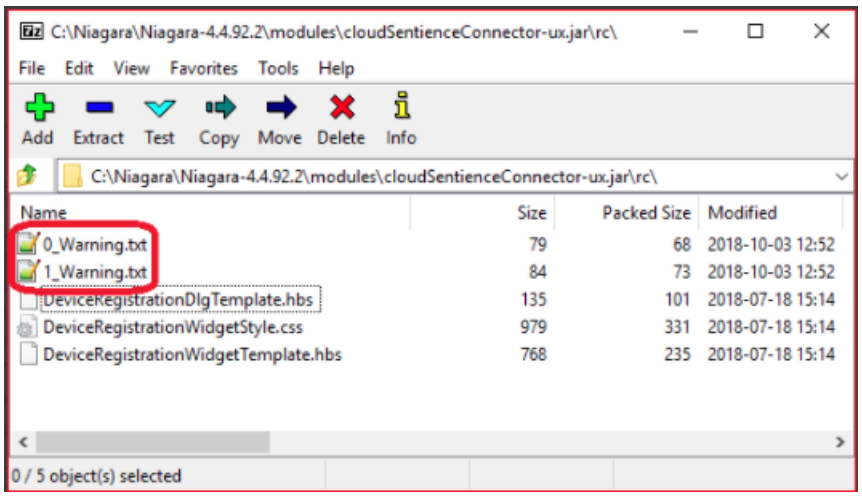
Step 1 In Windows in File Explorer, right-click on the module and click **7-Zip→Open archive**.

Step 2 In the 7-Zip **File Manager**, navigate into the `rc` folder.

A valid jar file will contain 5 files, including two JavaScript files, as shown.



A corrupted jar file will be missing these two Javascript files, and may potentially contain a text message file, stating that the potentially dangerous files have been removed, shown below.



Chapter 7 Troubleshooting

Topics covered in this chapter

♦ When an incident occurs
♦ Network sanity checks
♦ Troubleshooting

The Niagara CloudConnector and Niagara Cloud (NCHSD) can be difficult to troubleshoot when the communication is not optimal. This information is provided to make the troubleshooting and diagnosis of the connector and driver as straightforward as possible.

As new scenarios are encountered, this section will grow over time and become more comprehensive, but this is a starting list of issues that you may encounter, with information that may help to resolve them.

This troubleshooting information is intended for anyone who may be using the Niagara CloudConnector and NCHSD, or supporting those who are using it. The CloudConnector is also used in configuring Niagara stations to use Backup as a Service (NC BaaS), so the information on CloudConnector diagnosis is also relevant to users of NC BaaS, and support personnel for that product.

When an incident occurs

Collecting the information recommended below will help the Technical Support team get to the root cause of the problem quickly, characterize defects fully, and to address the problem for immediate and future users. If you can obtain it, the following information will be helpful to diagnose the problem.

What information to collect

- Date of incident
- Exact timestamp

  Timestamp from the host station output is fine unless it is known to be off.
- Customer or user in question, including brand
- Hardware platform (OS, version, etc.)
- Core Niagara software version (and any additional patches beyond base)
- Niagara Cloud modules versions - not just the release, but the specific version of each module

  This can be captured from the **Platform Administration** view, using the **View Details** button at the top.
- Any third party modules in use
- Product Family in question (NC BaaS, NCHSD)
- Any relevant log output or stack traces - see “What Logs to Collect”. More is better; extraneous information can be discarded if it is not important, but lost information cannot be recovered.
- Any relevant files - see “What Files to Collect”
- CloudConnector information - System ID, System Type, etc. see “Collecting CloudConnector Information”
- Steps taken before and after the problem? Be as specific and complete as possible. Information about the network environment is also critical in many cases - for example, is the host experiencing network disconnections? (either intentional or not).
- Steps taken to resolve the problem? Was the connector `disabled` or `enabled`, did you do an **Action→forceReconnect**, was the station restarted, did you attempt to reregister the connector, etc. Ideally, if the station state can be left unchanged, there may be steps that the Support team may suggest to correct the problem, or learn more about it.

CAUTION: Do not delete the CloudConnector and re-add it. If the connector has ever been successfully registered, deleting the connector results in a problem that can only be fixed by contacting Technical Support.

What logs to collect

There are several logs that can be enabled for diagnosing problems with the CloudConnector and NCHSD. The logs are listed in the following tables.

NOTE: If enabling moderately or highly verbose logs, stream the station output to a file to lessen the impact on performance. Also, saving the file gives you something to refer to later and share with Technical Support, if needed.

CloudConnector logs

Log Name	Description	Verbosity	Notes
`cloud.connector`	CloudConnector lifecyle events (connect ok/fail, disconnect, ping)	Low	set to FINE for debug
`cloud.connector.http`	CloudConnector HTTP information	Low	set to FINE for HTTP response codes
`cloud.connector.sentience`	Sentience-related connector information - authentication, heartbeat, token refresh	Moderate	set to FINE for all debug; set to CONFIG for some basic diagnostics
`cloud.iotmsg`	IoTHub Message Client information about message data and processing	Moderate to Very High*	set to FINE for basic diagnostics; FINEST for message payload (VERY verbose) ** In most cases where you are only interested in the message contents and not queuing issues, when setting `cloud.iotmsg` to FINEST, you will also want to set `cloud.iotmsg.queue` to INFO, as the logger levels inherit from their parent.
`cloud.iotmsg.queue`			set to FINER for most queue information *At the FINER/FINEST level, the large messages can fill the log very quickly

NCHSD logs

Log Name	Description	Verbosity	Notes
`ncloud`	General NCHSD (device lifecycle and initial message processing, pointId lookup)	Moderate	This is sort of a catch-all, it includes some alarm and command and security related messages
`ncloud.alarm`	Alarming information	Low	alarm message delivery
`ncloud.command`	Command processing	Moderate	Processing for System Commands (except point & cov)
`ncloud.history`	History export information, discovery/learn	Moderate	set to FINE for updates on history export execution
`ncloud.point`	Point information	High	Point update, read/write command processing, cov handling

Log Name	Description	Verbosity	Notes
ncloud.point.background	Background update group processing	Low	General execution of point batching (individual update info is in "ncloud.point")
ncloud.point.standard	Standard update group processing	Low	General execution of point batching (individual update info is in "ncloud.point")
ncloud.point.priority	Priority update group processing	Low	General execution of point batching (individual update info is in "ncloud.point")
ncloud.security	Information about command authorization and security	Moderate	Set to CONFIG to see role mapping **NOTE:** Information about Trust Store mapping is controled by "ncloud" log.
authentication	Information about user authentication in cloud commands	Low	User authentication; non-cloud, but may be useful in identifying failed command reason. See note following this table.

NOTE: The authentication log is protected by a property flag check. This is a hidden property in the AuthenticationService; you must unhide the property (in the **AX SlotSheet** view of the AuthenticationService, right-click **debug→Config Flags** and click to uncheck the **Hidden** flag).

IoTHub device client logs

There are a number of loggers within the Microsoft IoTHub Device Client that can be enabled to diagnose the IoTHub connection behavior in detail. If needed, you may be directed to enable them by Technical Support. It is not recommended to enable these loggers for ordinary users, so they are not listed here. Most of them begin with **com.microsoft.azure.sdk.iot.device**.

NOTE: If you are asked to enable these logs, remember to disable them by setting to INFO or higher levels as soon as the diagnostics session is completed. Otherwise they can produce large amounts of output that will significantly impact station performance.

Deciding which logs to enable

Choosing which logs to enable requires certain judgment on the part of the user. If you set every log to ALL, they will generate a huge amount of data which makes it harder to debug the problem. Each of the basic NCHSD functions (points, histories, alarms, commands) has a log level beginning with "**ncloud**". These do not generate a huge amount of data so, they can usually be set to ALL without negatively affecting performance.

- For issues with message security and authentication, set **ncloud.security** and **authentication** to ALL. The output level is usually low enough to be manageable.
- For issues with connection, registration, and connector behavior, set **cloud.connector**, **cloud.connector.http**, and **cloud.connector.sentience** to ALL, as they also generate relatively low output.
- For IoTHub concerns, we have a separate log, **cloud.iotmsg**. This is useful for examining specific message contents but can be extremely verbose, especially for a large system with many points and histories. This can be set to CONFIG or FINE, but use FINER or FINEST with care, and only for brief periods, as the output level is high. At the FINEST/ALL level, this provides the full contents of every message, such as histories. Only use this if the message contents are in question. At the FINE and FINER levels, it generates detailed information. At the CONFIG and INFO levels, it shows general information that can be helpful in tracing the message flow for other issues.

CAUTION: Do not forget to return logger settings to their default INFO levels once your problem is corrected. Leaving the loggers at higher levels of debug can impact system performance, and hide any new problems under a wave of noisy station output. This is especially true for the **cloud.iotmsg** logger.

What files to collect

After setting logs, collecting the station output is critical to diagnosing the problem. To do this, it is best to stream the station output to a file on your Workbench PC. This can be done from the **Application Director** window. If you are running from the Workbench User Home location, simply capture the text on the console window.

If the incident has already happened, it can be useful to go into the host's file system and get the older console output. This will be in the User Home with the name `console.txt`. Previous console logs from earlier station executions may also be useful. They will be listed under `console_backup_YYMMDD_HHMM.txt`. The station database is always helpful, and may allow Technical Support to determine configuration problems that lead to the behavior being investigated. This will be in the `config.bog` file, but it may also be helpful to include the full station using the station copier.

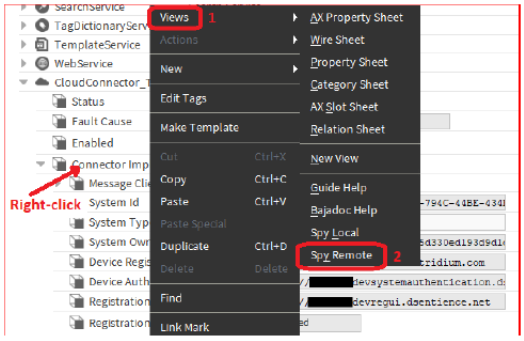
NOTE: If providing Technical Support with the bog file or full station copy, it's helpful to provide the username and password for the station.

Collecting CloudConnector information

This information is important if Support personnel need to make any modifications to the registration of the system in Asset Manager, or in the Honeywell Sentience Identity Provider. You can obtain most of this on the property sheet of the CloudConnector, with the Connector Impl expanded.

The CloudConnector's Id property is important, if it has been populated. Knowing that the Id is empty is also useful, so be sure to note if the Id is empty. The text field size often prevents full display of the values, so the best approach is to right-click on the Connector Impl, and select **Views→Spy Remote** (as shown), and then copy the entire text of that widow and paste it into a text file.

Figure 11 Collecting cloud connector information



Network sanity checks

If you are unable to register a JACE with Honeywell Sentience these sanity checks are intended to help you identify the source of the problem. Unfortunately on the JACE there is not the full spectrum of tools available to probe the network environment; however, all the basic checks below can be run from a JACE. If you can connect a laptop to the JACE network you will be able to run the tests in the additional checks section.

Basic checks

Basic sanity checks such as network port health, DNS health, and external communications can be run from a JACE.

Network port health

1. Log into the JACE through the USB port (see the *JACE-8000 Install and Startup Guide*) and enter `sh` to get a shell.
2. Run `ifconfig` to see if the JACE has an IP address. Output should be similar to that shown.

Figure 12 Network port health

```
1  $ ifconfig
2  lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33192
3          inet 127.0.0.1 netmask 0xff000000
4          inet6 ::1 prefixlen 128
5          inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
6  dm0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
7          address: 50:72:24:af:f7:e3
8          media: Ethernet autoselect (100baseTX full-duplex,flowcontrol)
9          status: active
10         inet 172.31.65.202 netmask 0xfffffc00 broadcast 172.31.67.255
11         inet6 fe80::5272:24ff:feaf:f7e3%dm0 prefixlen 64 scopeid 0x2
12 dm1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
13         address: 50:72:24:af:f7:e5
14         media: Ethernet none
15         inet 192.168.1.1 netmask 0xffffff00 broadcast 192.168.1.255
16         inet6 fe80::5272:24ff:feaf:f7e5%dm1 prefixlen 64 scopeid 0x3
17 pflog0: flags=0 mtu 33192
```

Line 6 starts the display of information about the primary network port and line 12 starts the display of the secondary network interface.

Line 8 indicates that the primary interface is currently connected where as line 14 indicates that the secondary interface is not connected. Lines 9 and 10 show the v4 and v6 IP address of the primary interface.

DNS health

To test that DNS is working you can attempt to ping your favorite web site and see if it is able to get an IP address for it.

NOTE: Most web sites will not respond to ping requests but for DNS testing you should get a response.

Figure 13 Using ping to test DNS health

```
1  $ ping www.google.com
2  PING www.google.com (216.58.216.4): 56 data bytes
```

In the example above we know DNS is working since *www.google.com* is resolved to an IP address, in this case `216.58.216.4`.

Below is an example where DNS lookup failed. Note that it usually takes a short time for the test to fail since the JACE needs to time-out waiting for the DNS server to respond.

Figure 14 DNS health2

```
1  $ ping www.google.com
2  ping: Cannot resolve "www.google.com" (Host name lookup failure)
```

External communication

To test that we can establish a secure connection to the outside world we can use serial shell (`ssh`) and attempt a connection to the Device Registration Service. For this test enter the host name of the Device Registration URL in your CloudConnector.

Figure 15 External communication

```
1  $ /usr/bin/ssh -v -p 443 niagara-cloud.com
2  OpenSSH_5.2 QNX_Secure_Shell-20090621, OpenSSL 1.0.2j  26 Sep 2016
3  debug1: Connecting to niagara-cloud.com [13.82.101.179] port 443.
4  debug1: Connection established.
5  Could not create directory '/.ssh'.
6  debug1: identity file /.ssh/identity type -1
7  debug1: identity file /.ssh/id_rsa type -1
8  debug1: identity file /.ssh/id_dsa type -1
```

You will need to use the fully qualified path to the `ssh` command. You will also need to specify the `-v` verbose flag to see what is happening.

Line 4 shows that we were able to successfully connect to the Device Registration Service. Since we are connecting to a web server and not a `sshd` server the connection will hang and you will need to enter `Ctrl+C` to get out of the command.

Additional checks

The following endpoint availability checks cannot be run from a JACE but if you have a Windows or Linux device attached to the same network as the JACE, or if you are installing a Supervisor this should provide some additional information.

Endpoint availability

Check to see if we can reach web endpoints required for device registration with a browser.

1. Go to **Device Registration Url→ping** (i.e. *https://devreg.cloud.tridium.com/ping*).

   This should return a JSON formatted response.

2. Go to **Device Authentication Url→api/authentication/rpkchallenge** (i.e. *https://gaprodsystemauthentication.sentience.honeywell.com/api/authentication/rpkchallenge*).

   This should return an XML formatted error message stating that the service does not support the GET method.

3. Go to **Registration Url→api/swagger/public** (i.e. *https://gaprodregui.sentience.honeywell.com/api/swagger/public*).

   This should return a JSON formatted response.

If you cannot reach the endpoints you can attempt to see where the problem is with the trace route (`tracert` on Windows) command, this will show the path through the network that packets are taking.

Figure 16 Endpoint availability for device registration

```
>tracert niagara-cloud.com

Tracing route to waws-prod-blu-075.api.niagara-cloud.com [13.82.101.179]
over a maximum of 30 hops:

  1     3 ms     3 ms     3 ms  137.19.60.3
  2     1 ms     1 ms     1 ms  137.19.35.237
  3    15 ms    16 ms    15 ms  10.160.16.2
  4    21 ms    15 ms    15 ms  10.223.255.229
  5    15 ms    15 ms    14 ms  10.223.255.65
  6    16 ms    16 ms    16 ms  10.223.255.58
  7    17 ms    15 ms    16 ms  10.221.192.36
  8    15 ms    15 ms    18 ms  199.64.6.87
  9    15 ms    15 ms    16 ms  199.64.6.52
 10    15 ms    16 ms    16 ms  199.64.6.77
 11    26 ms    58 ms    28 ms  12.249.243.109
 12    22 ms    22 ms    23 ms  cr2.phlpa.ip.att.net [12.123.237.142]
 13    24 ms    22 ms    22 ms  12.122.2.201
 14    22 ms    22 ms    22 ms  gar3.rcmva.ip.att.net [12.122.135.173]
 15    24 ms    20 ms    20 ms  12.122.135.109
 16    23 ms    27 ms    32 ms  12.247.95.62
 17    25 ms    24 ms    25 ms  be-74-0.ibr02.was05.ntwk.msn.net [104.44.9.42]
 18    24 ms    24 ms    24 ms  be-1-0.ibr01.was05.ntwk.msn.net [104.44.4.18]
 19    23 ms    23 ms    22 ms  be-5-0.ibr04.bl20.ntwk.msn.net [104.44.16.183]
 20    23 ms    23 ms    22 ms  ae161-0.icr01.bl7.ntwk.msn.net [104.44.21.230]
 21     *        *        *     Request timed out.
 22     *        *        *     Request timed out.
```

Entries that get a * represent network messages that timed out. If a host gets three * then it is either down or configured not to respond to ping traffic. Services running in the cloud are usually configured not to respond to ping traffic; however, we can see if our network traffic is making it out of the local network environment.

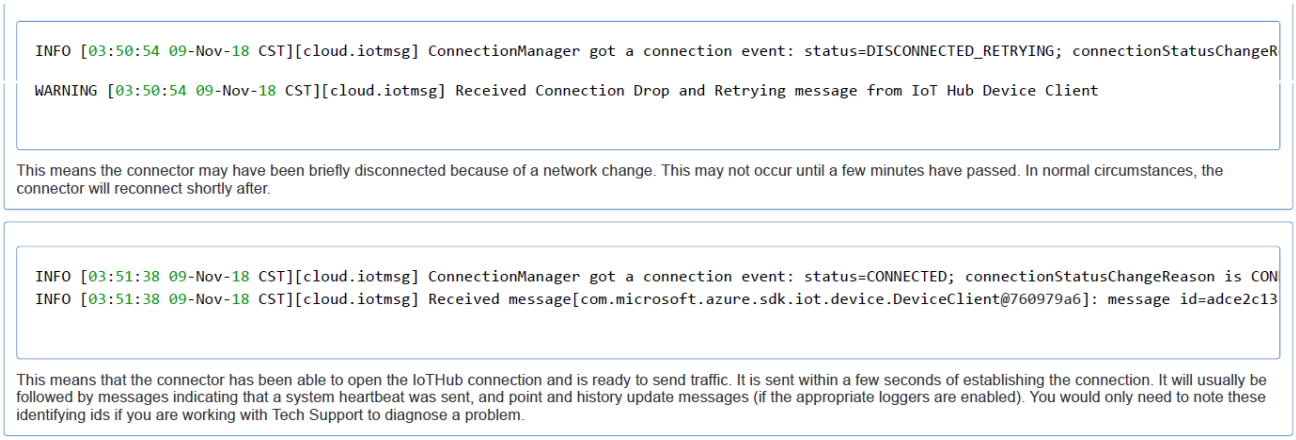
In the example above note that on lines 17 and 19 we get a response from a server owned by AT&T and then on lines 22 through 25 we are getting responses from Microsoft owned machines. This tells us that we were able to route out of the local environment onto the public internet.

We can also learn other things from these traces, for example if a host has one or two * on its line then that means the host or a host leading up to it is dropping packets. This will degrade performance and could lead to other problems. Also if there is a big jump in response times from one line to the next that could also be indicative of a potential network problem.

Normal log messages

There are several log messages reported by the CloudConnector and driver that are part of normal operation and are not a cause for concern, although they may appear confusing to those unfamiliar with the driver. Examples of normal log messages are shown below, along with an explanation of what they mean.

Figure 17 Normal log messages



Troubleshooting

A global enable flag, **`Batch Update Enabled`**, on the cloud point device extension controls all batch updates. This property can be useful to temporarily stop all traffic related to points for the purpose of diagnosing problems with another aspect of the station.

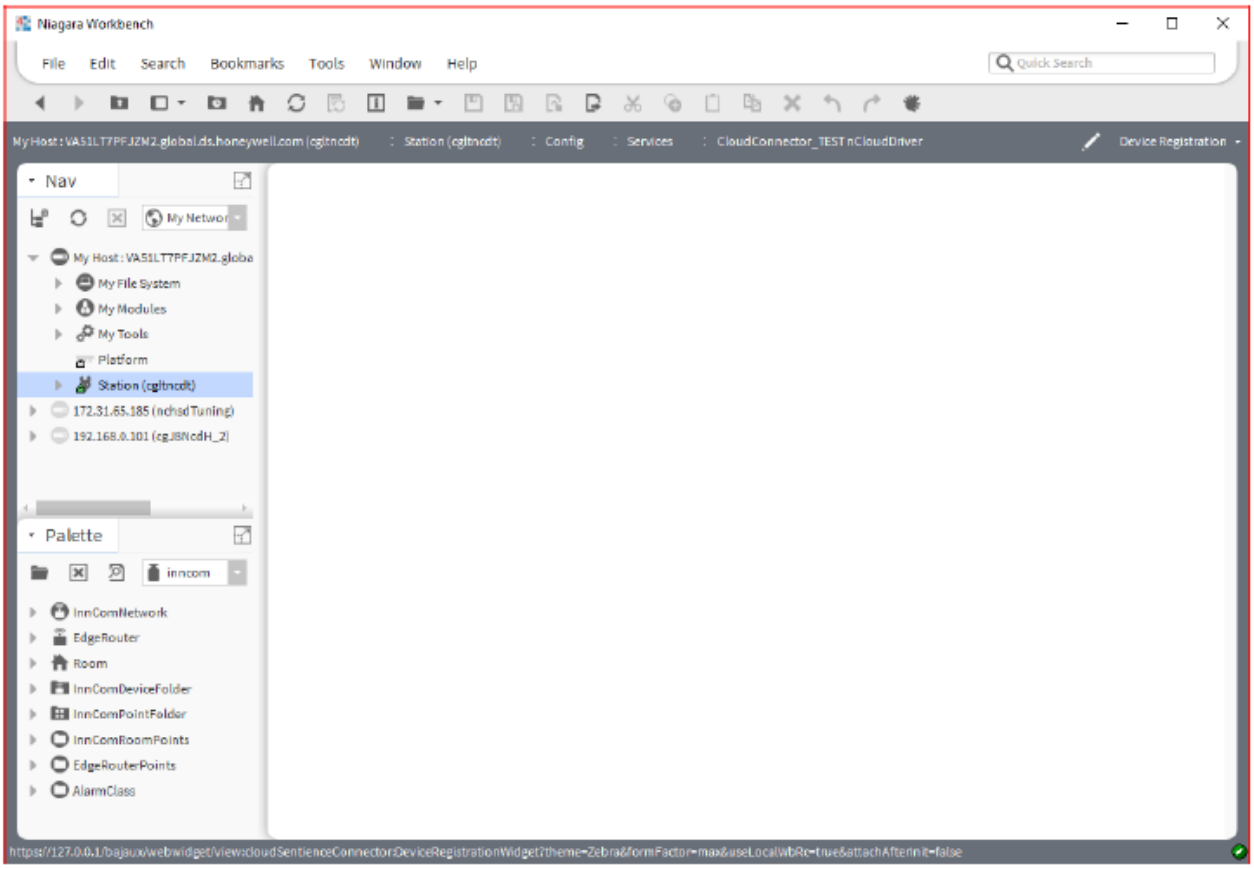
Device Registration view will not load

Troubleshooting tips for when the **Device Registration** view will not load.

Cause

The typical cause of this registration problem is a corrupted cloudSentienceConnector-ux module. This is evidenced by the “White Screen of Delay” seen after double-clicking the CloudConnector. This persistent blank view indicates that the **`Device Registration`** widget did not load.

Figure 18 Device Registration view does not load



Tips

- First, wait about 10-15 seconds and refresh the view. Sometimes the view presentation may take a little longer than expected to load, and can be resolved by a view refresh.
- Try resizing the main view pane by clicking the divider between the NavTree sidebar and the main view and drag the divider slightly to the left or right which causes the main view pane to "paint".
- Try restarting your browser. If using Workbench, close and restart it.
- If the widget still fails to load, try opening another **ux** view, such as the **User Manager** view of the User-Service, to see if that manager view loads successfully.
  - If the **User Manager** view does not load, there may be a more general problem.
  - If the **User Manager** view loads fine, you may have a problem with the **ux** module providing the **Device Registration** view. This could be because your cloudSentienceConnector-ux module has been corrupted, possibly due to IT algorithms stripping "potentially dangerous" files from within the jar file when emailing it to other users. To confirm that this is the problem, see "Confirming Niagara Modules" after delivery for details on how to tell if the module is affected.

Solution

If your cloudSentienceConnector-ux module is corrupted, you need to obtain a valid version of the module, either by downloading it directly from Niagara Community Software portal, or from a trusted source. For more information on how to distribute modules safely between users, see "Distributing NCHSD Modules Between Users" in the "Workflow" section of this guide.

Cannot reach device registration web service

Troubleshooting details for when the CloudConnector is prevented from reaching the Device Registraton web service.

Cause

This registration problem typically occurs when you are connected to the station using Workbench or a browser on a machine that does not have "sufficient access" to the internet which prevents you from performing device registration.

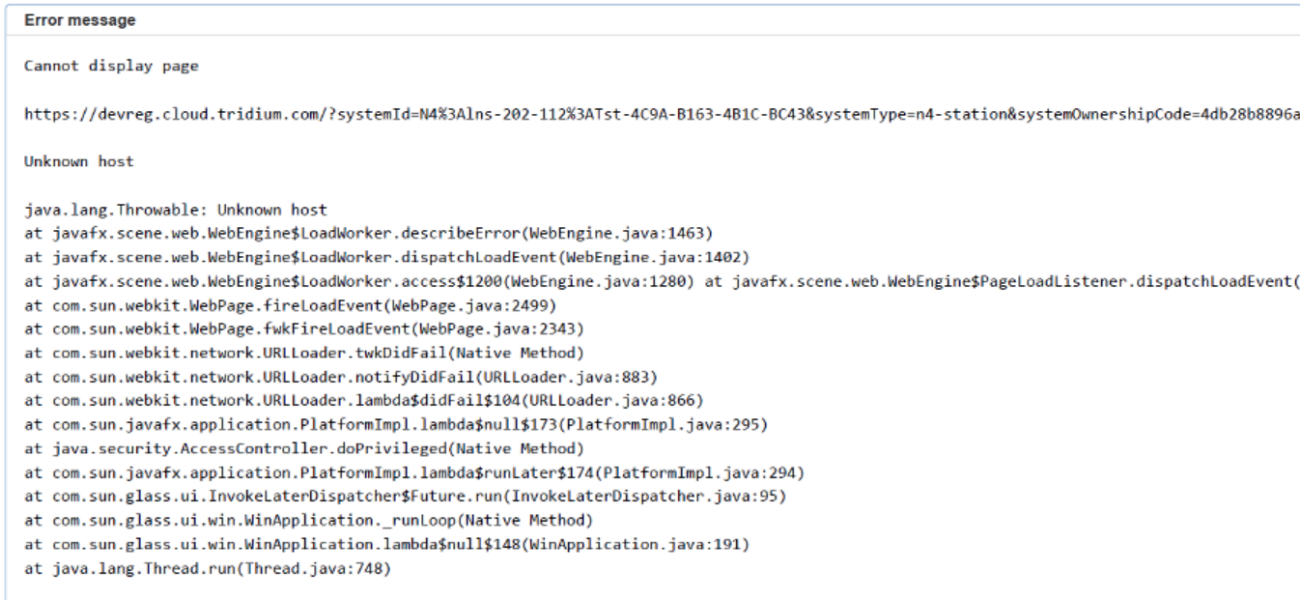
NOTE: The term “Sufficient access" means not only that the machine has access to the internet, but that certain proxy and firewall limitations are not in effect. For details, see the “Requirements” section in this guide. Your network configuration must satisfy the stated requirements.

Tip

You can confirm if your client Workbench or browser does not have internet access, or is blocked by proxy or firewall rules from reaching one of the necessary destinations, if you see the following error (or a similarly worded error):

NOTE: This would be seen in the Workbench VM, not the station VM.

Figure 19 Error message



Solution

Ensure that your client machine running Workbench or the browser has internet access before attempting device registration. Also, make sure that the URLs specified in the “Requirements” section of this guide are accessible to the client machine, and are not blocked by a network proxy or firewall configuration.

NOTE: It is also necessary for the station host to have internet access satisfying the requirements because it must authenticate directly with the identity provider to enable cloud communication.

Honeywell Sentience cloud platform connection issue

If you see the following warning on the station output, then your CloudConnector is not connecting to the Honeywell Sentience cloud platform.

Figure 20 Station output warning

```
WARNING [14:41:57 02-Oct-18 EDT][cloud.connector] Cannot connect to Cloud
```

To help with troubleshooting this set the **cloud.connector.http** and **cloud.connector.sentience** log levels to FINE or a lower level (such as FINER, FINEST, ALL). Also, you may need to adjust some additional log levels such as **cloud.iotmsg**.

The first step in the troubleshooting process is to confirm the device status with the Honeywell Sentience Operations team. Once it is confirmed that the device is enabled, further diagnosis can proceed.

Figure 21 System disabled by Honeywell Forge Operations

System Disabled

The Honeywell Forge Operations team tracks and aggressively manages the bandwidth usage of systems participating in the Cloud Platform ecosystem. Devices that send too much data are subject to being disabled from the Sentience IoTHub. This means that the device is prevented from sending any data to the Sentience IoTHub. The device may even be prevented from establishing the IoTHub connection in the first place. This may manifest in several different ways. One example is where the connector is able to authenticate to the identity endpoint, but cannot open the connection. You may see the Property Sheet of the Cloud Connector show "Connected", or possibly "Pending Connect". The following message, or something similar, may show in the station output:

```
1
2  CONFIG [14:33:56 29-Jul-19 BST][cloud.connector.sentience] Starting RPK Challenge
3  CONFIG [14:33:56 29-Jul-19 BST][cloud.connector.sentience] Sending RPK Challenge request to URI https://gaprodsystemauthentication.se
4  FINE [14:33:57 29-Jul-19 BST][cloud.connector.http] HTTP Response Code:200
5  CONFIG [14:33:57 29-Jul-19 BST][cloud.connector.sentience] Checking for existing locally initialized keys
6  CONFIG [14:33:58 29-Jul-19 BST][cloud.connector.sentience] Authenticating using software keys
7  CONFIG [14:33:58 29-Jul-19 BST][cloud.connector.sentience] Sending RPK Challenge Response to URI https://gaprodsystemauthentication.s
8  FINE [14:33:59 29-Jul-19 BST][cloud.connector.http] HTTP Response Code:200
9  CONFIG [14:33:59 29-Jul-19 BST][cloud.connector.sentience] Completed RPK Challenge - 2438 ms
10 CONFIG [14:33:59 29-Jul-19 BST][cloud.connector.sentience] Starting System Connections
11 WARNING [14:33:59 29-Jul-19 BST][com.microsoft.azure.sdk.iot.device.transport.amqps.AmqpsDeviceAuthenticationCBSTokenRenewalTask] jav
12 FINE [14:34:02 29-Jul-19 BST][cloud.connector.http] HTTP Response Code:200
13 CONFIG [14:34:02 29-Jul-19 BST][cloud.connector.sentience] Completed System Connections: 3125 ms
14 FINEST [14:34:03 29-Jul-19 BST][cloud.connector] BCloudConnector.pingFail(Could not open the connection), notifying connectCallbacks
15 FINE [14:34:03 29-Jul-19 BST][cloud.connector] Connection fail
16 java.io.IOException: Could not open the connection
17     at com.microsoft.azure.sdk.iot.device.DeviceIO.open(DeviceIO.java:165)
18     at com.microsoft.azure.sdk.iot.device.DeviceClient.open(DeviceClient.java:369)
19     at com.tridium.cloud.client.iotdep.BIotHubMessageClient.lambda$onConnect$4(BIotHubMessageClient.java:441)
20     at java.security.AccessController.doPrivileged(Native Method)
21     at com.tridium.cloud.client.iotdep.BIotHubMessageClient.onConnect(BIotHubMessageClient.java:387)
22     at com.tridium.cloud.client.iothub.BAbstractIotHubConnectorImpl.doConnect(BAbstractIotHubConnectorImpl.java:79)
23     at com.tridium.cloud.client.sentience.BSentienceConnectorImpl.doConnect(BSentienceConnectorImpl.java:734)
24     at com.tridium.cloud.client.BConnectorImpl.connect(BConnectorImpl.java:118)
25     at com.tridium.cloud.client.BCloudConnector.reconnectSync(BCloudConnector.java:527)
26     at java.util.concurrent.FutureTask.run(FutureTask.java:266)
27     at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$201(ScheduledThreadPoolExecutor.java:180)
28     at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:293)
29     at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
30     at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
31     at java.lang.Thread.run(Thread.java:748)
32 Caused by: com.microsoft.azure.sdk.iot.device.exceptions.TransportException: Unknown transport exception occurred
33     at com.microsoft.azure.sdk.iot.device.transport.amqps.AmqpsIotHubConnection.onLinkRemoteClose(AmqpsIotHubConnection.java:729)
34     at org.apache.qpid.proton.engine.BaseHandler.handle(BaseHandler.java:176)
35     at org.apache.qpid.proton.engine.impl.EventImpl.dispatch(EventImpl.java:108)
36     at org.apache.qpid.proton.reactor.impl.ReactorImpl.dispatch(ReactorImpl.java:324)
37     at org.apache.qpid.proton.reactor.impl.ReactorImpl.process(ReactorImpl.java:291)
38     at com.microsoft.azure.sdk.iot.device.transport.amqps.IotHubReactor.run(IotHubReactor.java:28)
39     at com.microsoft.azure.sdk.iot.device.transport.amqps.AmqpsIotHubConnection$ReactorRunner.call(AmqpsIotHubConnection.java:824)
40     at java.util.concurrent.FutureTask.run(FutureTask.java:266)
       ... 3 more
```

NOTE: In the above image, the authentication steps all return an HTTP 200 (see lines 3, 7, and 11), indicating success. The exception occurs only when attempting to establish the IoTHub connection (see lines 32–40).

Solution

If this is happening, your device may be blocked (i.e., throttled) by Honeywell Sentience due to sending too much traffic at some point. If that is the case, you should have received an email from the Honeywell Sentience Operations team indicating that the device has been blocked.

- If you did receive such an email, contact your Honeywell Support channel to request that your device's ability to connect with Honeywell Sentience be re-enabled.
- If you did not receive such an email, then there may be a problem with the email addresses on file for this system. You may need to work with your Honeywell Support channel to set the proper notification configurations.

If you are experiencing throttling, you will need to update to the latest version of the NCHSD software, including the CloudConnector modules. The corresponding modules for each release are listed in the following table.

Niagara Release	NCHSD Release Needed	Cloud Module Versions
Niagara 4.4	NCHSD 1.3.1	2018.5.11* *cloudIotHubConnector-rt and cloudSentienceConnector-rt are at version 2018.5.11.1, due to patches
Niagara 4.8	NCHSD 4.8.0.1	2019.1 These NCHSD modules are used in Niagara 4.8 and in FC 1.1
Niagara 4.9	NCHSD 4.9.0.2	2019.2 These NCHSD modules are used in Niagara 4.9 and in FC 1.2
Niagara 4.10	NCHSD 4.10.0.1	2020.1 These NCHSD modules are used in Niagara 4.10 and in FC 1.3

NOTE: You should adjust your backfill protection settings, as described in "nCloudDriver-CloudHistoryDeviceExt" section of this guide, under the heading "Cloud histories and restored connectivity". If you do not have any backfill properties in your History Device Extension, then you definitely need to upgrade to a newer version of NCHSD software.

General connection issues

There are many different reasons why the CloudConnector might not be able to send data to Honeywell Sentience through the IoTHub. Many of them relate to issues outside of the driver itself. This section lists a few common symptoms, along with potential causes and remedies.

Connector is not registered

If your station output appears as shown:

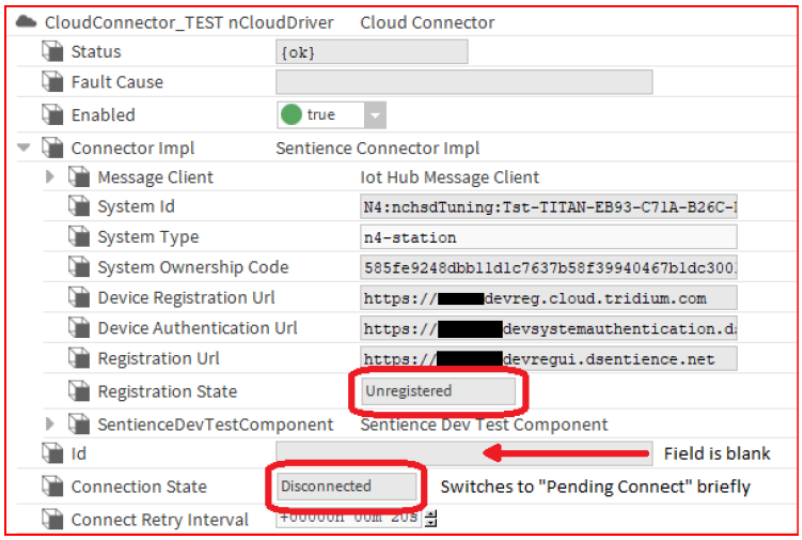
Figure 22 Station output

```
CONFIG [14:55:54 02-Oct-18 EDT][cloud.connector.sentience] Connecting to Sentience
CONFIG [14:55:57 02-Oct-18 EDT][cloud.connector.sentience] Starting RPK Challenge
CONFIG [14:55:57 02-Oct-18 EDT][cloud.connector.sentience] Sending RPK Challenge request to URI https://gxxxxxdevsystemauthentication.dsent
FINE [14:55:58 02-Oct-18 EDT][cloud.connector.http] HTTP Response Code:404
WARNING [14:55:58 02-Oct-18 EDT][cloud.connector] Cannot connect to Cloud
```

The HTTP 404 "Not Found" response in the station output shows that the Honeywell Sentience Identity Provider does not have a record of this system.

And, if your CloudConnector properties appear as shown here:

Figure 23 CloudConnector properties for unsuccessful connection



- **`Registration State`**: `Unregistered`
- **`Id`**: blank (the connector has never found its **`System Guid`** by connecting to Honeywell Sentience)
- **`Connection State`**: `Disconnected`

Of course, this indicates that the CloudConnector is not registered.

Solution

Register the device with Niagara Cloud and Honeywell Sentience by opening the **Device Registration** view of the CloudConnector. Click **Register Device**, and follow the on-screen prompts as described in the "Install and configure" section of this guide, under the heading "Registering the device".

Connector keys are lost

This applies to all non-QNX stations, and to JACE-8000 hosts running Niagara 4.4. Controllers running Niagara 4.6 or later should use hardware encryption, **unless** the CloudConnector was originally registered under Niagara 4.4 or earlier.

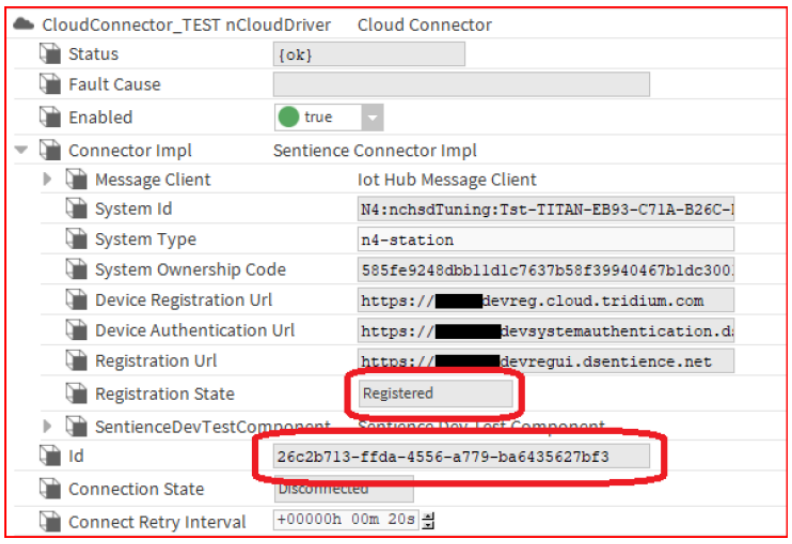
If your station output appears as shown here:

Figure 24 Station output

```
CONFIG [15:29:00 02-Oct-18 EDT][cloud.connector.sentience] Connecting to Sentience
CONFIG [15:29:03 02-Oct-18 EDT][cloud.connector.sentience] Starting RPK Challenge
CONFIG [15:29:03 02-Oct-18 EDT][cloud.connector.sentience] Sending RPK Challenge request to URI https://gxxxxxxdevsystemauthentication.dsentience.net/a
FINE [15:29:04 02-Oct-18 EDT][cloud.connector.http] HTTP Response Code:200
CONFIG [15:29:04 02-Oct-18 EDT][cloud.connector.sentience] Sending RPK Challenge Response to URI https://gxxxxxxdevsystemauthentication.dsentience.net/
FINE [15:29:04 02-Oct-18 EDT][cloud.connector.http] HTTP Response Code:401
WARNING [15:29:04 02-Oct-18 EDT][cloud.connector] Cannot connect to Cloud
```

And, your CloudConnector properties appear as shown here:

Figure 25 CloudConnector properties for lost connector keys



This indicates that the station is registered with Honeywell Sentience, but it cannot authenticate. Most likely your station does not have the required public/private key pair necessary to authenticate to Honeywell Sentience. You can confirm this by checking the **Certificate Manager** for a key with an alias matching the station name (note the alias will be all lowercase, with hyphens replacing any underscores in the station name). If you do not see the alias for your station, then your station cannot register because it does not have the necessary key pair.

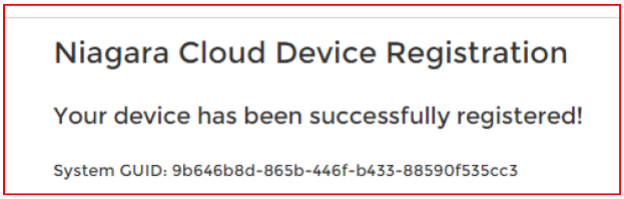
Solution

If you have saved the keys using the steps described in the "Upgrading a Host" section, you can try importing the key back into your **User Key Store** in the **Certificate Manager**. If the file contains the correct keys, the CloudConnector should be able to reconnect successfully.

Otherwise, you will need to deregister your system with Honeywell Sentience and re-register it. This currently requires you to contact your Support channel, and request that they de-registering your system.

Connector does not attempt connection

If you have registered your CloudConnector and received the following successfully registered notification, but your CloudConnector does not attempt to connect at all (i.e., there is no "Connecting to Sentience" message in the station output, when you have **`cloud.connector.sentience`** log set to ALL).



This problem may be more likely to occur if the **`System Type`** field had to be manually entered because the CloudConnector was added from the **`cloudSentienceConnector`** palette instead of from one of the branded palettes.

Solution

Try disabling the CloudConnector, and then re-enabling it.

Certificate validation issue

Old cloudConfig-rt module

If your station output shows the following error you may be using an older version of the cloudConfig-rt module, which specifies an outdated certificate for the Honeywell Sentience authentication endpoint.

Figure 26 Station output shows "Cannot connect to Cloud"

```
WARNING [13:31:40 28-Sep-18 PDT][cloud.connector] Cannot connect to Cloud
javax.baja.sys.ActionInvokeException
 at com.tridium.sys.schema.ComponentSlotMap.invoke(ComponentSlotMap.java:1904)
 at com.tridium.sys.schema.ComponentSlotMap.invoke(ComponentSlotMap.java:1853)
 at javax.baja.sys.BComponent.invoke(BComponent.java:1230)
 at com.tridium.cloud.client.sentience.BSentienceConnectorImpl.validateCertificate(BSentienceConnectorImpl.java:1306)
 at com.tridium.cloud.client.sentience.BSentienceConnectorImpl.doConnect(BSentienceConnectorImpl.java:657)
 at com.tridium.cloud.client.BConnectorImpl.connect(BConnectorImpl.java:118)
 at com.tridium.cloud.client.BCloudConnector.reconnectSync(BCloudConnector.java:527)
 at java.util.concurrent.FutureTask.run(FutureTask.java:266)
 at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$201(ScheduledThreadPoolExecutor.java:180)
 at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:293)
 at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
 at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
 at java.lang.Thread.run(Thread.java:748)
Caused by: javax.security.cert.CertificateExpiredException: certificate expired on 20180802120000GMT+00:00
 at com.sun.security.cert.internal.x509.X509V1CertImpl.checkValidity(X509V1CertImpl.java:186)
 at com.sun.security.cert.internal.x509.X509V1CertImpl.checkValidity(X509V1CertImpl.java:168)
 at com.tridium.cloud.client.BCertObject.doExtractPublicKey(BCertObject.java:72)
 at auto.com_tridium_cloud_client_BCertObject.invoke(AutoGenerated)
 at com.tridium.sys.schema.ComponentSlotMap.invoke(ComponentSlotMap.java:1888)
 ... 12 more
```

This indicates that the station does not accept the validity of the certificate provided by the authentication endpoint. To confirm if this is the case:

1. Expand the **`CloudConnector`**, right-click the **`Connector Impl`**, and click **Views→Spy Remote**.
2. In the **Spy** view, scroll to the **Properties** section and see if the **`cloudCertVerification`** property is `false`. If it is `true`, you likely have an outdated module.

Solution

Ensure that you have the latest released version of all of the NCHSD modules so that you will not experience certificate validation issues with accepting the validity of the authentication endpoint.

- For Niagara 4.6.96.28 and older, use the NCHSD modules versioned 2018.6.7. You can get the modules from the Niagara 4.7 release image (they have Niagara 4.6 dependencies).
- For Niagara 4.4.93.40 and older, use the cloudSentienceConnector-rt 2018.4.2.1 patch, with the other NCHSD modules versioned 2018.4.2.

Network environment issues

The station does not know it is registered

Once the **Device Registratio**n widget on the station hyperlinks you to the remote **Device Registration** web service, the registration process occurs on the remote site, not in the station. The station knows nothing about the subsequent steps in that process.

So, when you register the device's Host Id with the Asset Manager, and device's System Id with the Honeywell Sentience Identity Provider, the station is unaware of this.

The only process occurring in the station is the RPK Authentication, and that occurs only once the Honeywell Sentience registration process completes. Once that is true, the RPK Authentication process can succeed. The station will obtain the correct Registration status from Honeywell Sentience, and establish connection to the IoTHub.

Proxy server preventing connection

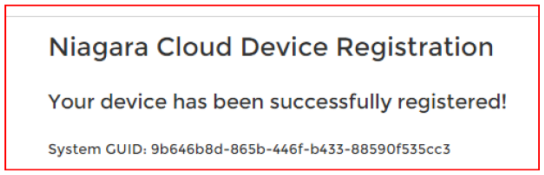
If you are able to register the station with Honeywell Sentience and Niagara Cloud, but the station cannot connect (the connector's Connection State never gets to "Connected"), then there may be a problem with the local IT network's proxy settings, or with the firewall settings imposed upon the station.

NOTE: The NCHSD CloudConnector requires Unauthenticated Proxy Access. Without this, you will be asked to provide credentials to the proxy server, and to approve exemptions into the Niagara **User Trust Store**. This process will repeat itself frequently and does not provide a workable solution.

NOTE: CloudConnector only allows Transparent Proxy Server. Prior to Niagara 4.9, the NCHSD CloudConnector requires that any proxy server used to connect to the outside internet be a Transparent Proxy Server. Named Proxy Servers (also known as "Explicit Proxy Servers") are not supported.

If network settings prevent the station from connecting properly, the station will remain in the Unregistered state, even if it is registered. If you have gotten the successful registration notification (shown) your device's System Id is successfully registered with Honeywell Sentience and Niagara Cloud. If the device's Cloud Connector is still showing Unregistered, it is most likely because it cannot reach the authentication endpoint.

Figure 27 Successfully registered message



If the station output shows messages similar to the following, then your station may be behind a named (explicit) proxy server. This configuration is not supported prior to Niagara 4.9.

Figure 28 Station output for station behind named proxy server

```
FINE [13:01:20 08-May-19 CDT][crypto] hostname didn't verify
java.security.cert.CertificateException: hostname didn't verify
at com.tridium.crypto.core.cert.TridiumX509TrustManager.checkServerTrusted(Unknown Source)
at sun.security.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:1496)
at sun.security.ssl.ClientHandshaker.processMessage(ClientHandshaker.java:216)
at sun.security.ssl.Handshaker.processLoop(Handshaker.java:1026)
at sun.security.ssl.Handshaker.process_record(Handshaker.java:961)
at sun.security.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:1072)
at sun.security.ssl.SSLSocketImpl.performInitialHandshake(SSLSocketImpl.java:1385)
at sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:1413)
at sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:1397)
at sun.net.www.protocol.https.HttpsClient.afterConnect(HttpsClient.java:559)
at sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.connect(AbstractDelegateHttpsURLConnection.j
at sun.net.www.protocol.http.HttpURLConnection.getOutputStream0(HttpURLConnection.java:1334)
at sun.net.www.protocol.http.HttpURLConnection.access$100(HttpURLConnection.java:91)
at sun.net.www.protocol.http.HttpURLConnection$8.run(HttpURLConnection.java:1301)
at sun.net.www.protocol.http.HttpURLConnection$8.run(HttpURLConnection.java:1299)
at java.security.AccessController.doPrivileged(Native Method)
at java.security.AccessController.doPrivilegedWithCombiner(AccessController.java:782)
at sun.net.www.protocol.http.HttpURLConnection.getOutputStream(HttpURLConnection.java:1298)
at sun.net.www.protocol.https.HttpsURLConnectionImpl.getOutputStream(HttpsURLConnectionImpl.java:259)
at com.tridium.cloud.util.StandardHttpUtils.makeConnection(StandardHttpUtils.java:406)
at com.tridium.cloud.util.StandardHttpUtils.lambda$http$0(StandardHttpUtils.java:240)
at java.security.AccessController.doPrivileged(Native Method)
at com.tridium.cloud.util.StandardHttpUtils.http(StandardHttpUtils.java:237)
at com.tridium.cloud.util.StandardHttpUtils.post(StandardHttpUtils.java:165)
at com.tridium.cloud.client.sentience.BSentienceConnectorImpl.rpkChallenge(BSentienceConnectorImpl.java:1724)
at com.tridium.cloud.client.sentience.BSentienceConnectorImpl.registerDevice(BSentienceConnectorImpl.java:891
at com.tridium.cloud.client.sentience.BSentienceConnectorImpl.doConnect(BSentienceConnectorImpl.java:724)
at com.tridium.cloud.client.BConnectorImpl.connect(BConnectorImpl.java:118)
at com.tridium.cloud.client.BCloudConnector.reconnectSync(BCloudConnector.java:527)
```

```
at java.util.concurrent.FutureTask.run(FutureTask.java:266)
at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$201(ScheduledThreadPoolExecutc
at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
at java.lang.Thread.run(Thread.java:748)

SEVERE [13:01:20 08-May-19 CDT][crypto] No trusted certificate found
SEVERE [13:01:20 08-May-19 CDT][crypto] certificate private key has changed
FINEST [13:01:20 08-May-19 CDT][cloud.connector] BCloudConnector.pingFail(java.security.cert.CertificateExcep
FINE [13:01:20 08-May-19 CDT][cloud.connector] Connection fail
javax.net.ssl.SSLHandshakeException: java.security.cert.CertificateException: certificate private key has cha
at sun.security.ssl.Alerts.getSSLException(Alerts.java:192)
at sun.security.ssl.SSLSocketImpl.fatal(SSLSocketImpl.java:1959)
at sun.security.ssl.Handshaker.fatalSE(Handshaker.java:302)
at sun.security.ssl.Handshaker.fatalSE(Handshaker.java:296)
at sun.security.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:1514)
at sun.security.ssl.ClientHandshaker.processMessage(ClientHandshaker.java:216)
at sun.security.ssl.Handshaker.processLoop(Handshaker.java:1026)
at sun.security.ssl.Handshaker.process_record(Handshaker.java:961)
at sun.security.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:1072)
at sun.security.ssl.SSLSocketImpl.performInitialHandshake(SSLSocketImpl.java:1385)
at sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:1413)
at sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:1397)
at sun.net.www.protocol.https.HttpsClient.afterConnect(HttpsClient.java:559)
at sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.connect(AbstractDelegateHttpsURLConnection.j
at sun.net.www.protocol.http.HttpURLConnection.getOutputStream0(HttpURLConnection.java:1334)
```

```
at sun.net.www.protocol.http.HttpURLConnection.access$100(HttpURLConnection.java:91)
at sun.net.www.protocol.http.HttpURLConnection$8.run(HttpURLConnection.java:1301)
at sun.net.www.protocol.http.HttpURLConnection$8.run(HttpURLConnection.java:1299)
at java.security.AccessController.doPrivileged(Native Method)
at java.security.AccessController.doPrivilegedWithCombiner(AccessController.java:782)
at sun.net.www.protocol.http.HttpURLConnection.getOutputStream(HttpURLConnection.java:1298)
at sun.net.www.protocol.https.HttpsURLConnectionImpl.getOutputStream(HttpsURLConnectionImpl.java:259)
at com.tridium.cloud.util.StandardHttpUtils.makeConnection(StandardHttpUtils.java:406)
at com.tridium.cloud.util.StandardHttpUtils.lambda$http$0(StandardHttpUtils.java:240)
at java.security.AccessController.doPrivileged(Native Method)
at com.tridium.cloud.util.StandardHttpUtils.http(StandardHttpUtils.java:237)
at com.tridium.cloud.util.StandardHttpUtils.post(StandardHttpUtils.java:165)
at com.tridium.cloud.client.sentience.BSentienceConnectorImpl.rpkChallenge(BSentienceConnectorImpl.java:1724)
at com.tridium.cloud.client.sentience.BSentienceConnectorImpl.registerDevice(BSentienceConnectorImpl.java:891
at com.tridium.cloud.client.sentience.BSentienceConnectorImpl.doConnect(BSentienceConnectorImpl.java:724)
at com.tridium.cloud.client.BConnectorImpl.connect(BConnectorImpl.java:118)
at com.tridium.cloud.client.BCloudConnector.reconnectSync(BCloudConnector.java:527)
at java.util.concurrent.FutureTask.run(FutureTask.java:266)
at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$201(ScheduledThreadPoolExecutc
at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
at java.lang.Thread.run(Thread.java:748)
Caused by: java.security.cert.CertificateException: certificate private key has changed
at com.tridium.crypto.core.cert.TridiumX509TrustManager.handleRejectedCertificate(Unknown Source)
at com.tridium.crypto.core.cert.TridiumX509TrustManager.handleCertException(Unknown Source)
at com.tridium.crypto.core.cert.TridiumX509TrustManager.checkServerTrusted(Unknown Source)
at sun.security.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:1496)
... 33 more
```

Solution

Review with your IT administrator the section "Setting up device internet access" in the “Getting Started” section of this guide, specifically regarding firewall access and unauthenticated transparent proxy access. This is a common problem with network setup, especially in a heavily restricted corporate or educational network. Ensure that the requirements in this section are met by the IT network configuration. Note that the transparent proxy requirement was added only in very recent versions of the documentation, so it may not be present in the documentation delivered with the software. For the most up-to-date version of the documentation, visit *https://docs.niagara-community.com/*.

If your network is using a proxy server, you may be able to find success by including the following lines into your `system.propertes` file, usually found in `/opt/niagara` on JACE controllers. Use the **File Transfer Client** to retrieve the file from the JACE to your Workbench and add the lines in a text editor, then copy the modified `system.propertes` file back to the JACE using the **File Transfer Client**. You should also apply the same changes to the `system.properties` file on your Workbench, if it is also subject to the same proxy server, as the Workbench will be used in registering the device.

Figure 29 Proxy settings

Proxy Settings

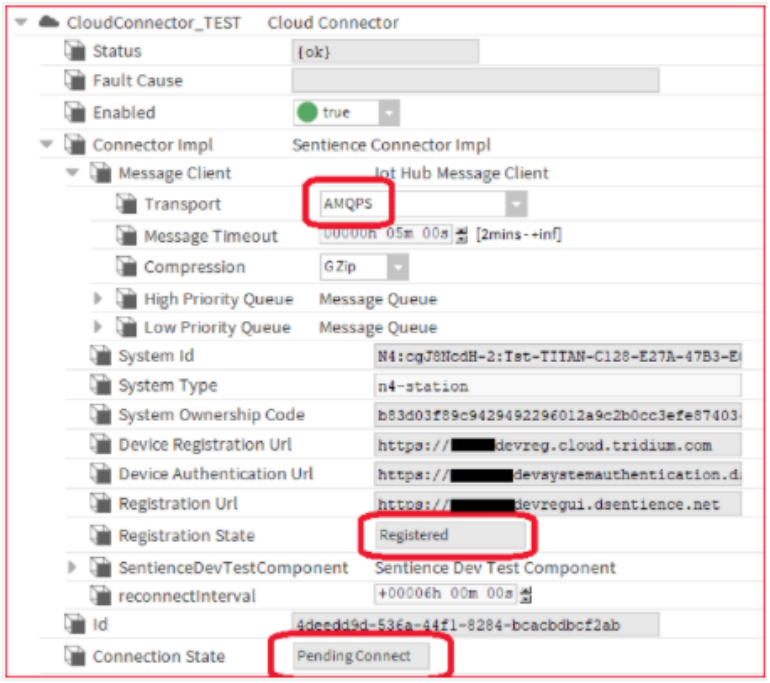
```
http.proxyHost=nnn.nnn.nnn.nnn          # HTTP Proxy Server address
http.proxyPort=nnnn                     # HTTP Proxy Server port
http.nonProxyHosts=nnn.nnn.nnn.nnn      # Put JACE IP address here
https.proxyHost=nnn.nnn.nnn.nnn         # HTTPS Proxy Server address
https.proxyPort=nnnn                    # HTTPS Proxy Server port
https.nonProxyHosts=nnn.nnn.nnn.nnn     # Put JACE IP address here
```

If your local network is using a named proxy server, you will need to work with your network administrator to obtain transparent proxy server access.

AMQPS Blocked

If you're using AMQPS as your Transport setting, you've registered your CloudConnector, and you're seeing the connector status stuck in "Pending Connect", it may be because AMQPS is blocked on your network.

Figure 30 CloudConnector properties for AMQPS blocked



This can be confirmed by looking at the station output (as a connection problem, having the **`cloud.connector`**, **`cloud.connector.http`**, and **`cloud.connector.sentience`** logs set to ALL will help here.)

Figure 31 Station output for AMQPS blocked

```
FINE [09:11:07 04-Oct-18 EDT][cloud.connector] BCloudConnector reconnectSync exception:java.io.IOException: Could not open the connection; connectRetryFuture:java.uti
CONFIG [09:11:07 04-Oct-18 EDT][cloud.connector.sentience] Connecting to Sentience
CONFIG [09:11:11 04-Oct-18 EDT][cloud.connector.sentience] Starting RPK Challenge
CONFIG [09:11:11 04-Oct-18 EDT][cloud.connector.sentience] Sending RPK Challenge request to URI https://gxxxxxxdevsystemauthentication.dsentience.net/api/authenticati
FINE [09:11:11 04-Oct-18 EDT][cloud.connector.http] HTTP Response Code:200
CONFIG [09:11:11 04-Oct-18 EDT][cloud.connector.sentience] Sending RPK Challenge Response to URI https://gxxxxxxdevsystemauthentication.dsentience.net/api/authenticat
FINE [09:11:13 04-Oct-18 EDT][cloud.connector.http] HTTP Response Code:200
CONFIG [09:11:13 04-Oct-18 EDT][cloud.connector.sentience] Completed RPK Challenge - 1773 ms
CONFIG [09:11:13 04-Oct-18 EDT][cloud.connector.sentience] Starting System Connections
FINE [09:11:13 04-Oct-18 EDT][cloud.connector.http] HTTP Response Code:200
CONFIG [09:11:13 04-Oct-18 EDT][cloud.connector.sentience] Completed System Connections: 815 ms
FINEST [09:12:28 04-Oct-18 EDT][cloud.connector] BCloudConnector.pingFail(Could not open the connection), notifying connectCallbacks
FINE [09:11:07 04-Oct-18 EDT][cloud.connector] Connection fail
java.io.IOException: Could not open the connection
 at com.microsoft.azure.sdk.iot.device.DeviceIO.open(DeviceIO.java:165)
 at com.microsoft.azure.sdk.iot.device.DeviceClient.open(DeviceClient.java:369)
 at com.tridium.cloud.client.iotdep.BIotHubMessageClient.lambda$onConnect$4(BIotHubMessageClient.java:441)
 at java.security.AccessController.doPrivileged(Native Method)
 at com.tridium.cloud.client.iotdep.BIotHubMessageClient.onConnect(BIotHubMessageClient.java:387)
 at com.tridium.cloud.client.iothub.BAbstractIotHubConnectorImpl.doConnect(BAbstractIotHubConnectorImpl.java:79)
 at com.tridium.cloud.client.sentience.BSentienceConnectorImpl.doConnect(BSentienceConnectorImpl.java:639)
 at com.tridium.cloud.client.BConnectorImpl.connect(BConnectorImpl.java:118)
 at com.tridium.cloud.client.BCloudConnector.reconnectSync(BCloudConnector.java:527)
 at java.util.concurrent.FutureTask.run(FutureTask.java:266)
 at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$201(ScheduledThreadPoolExecutor.java:180)
 at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:293)
 at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
 at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
 at java.lang.Thread.run(Thread.java:748)
Caused by: com.microsoft.azure.sdk.iot.device.transport.amqps.exceptions.AmqpConnectionFramingErrorException
 at com.microsoft.azure.sdk.iot.device.transport.amqps.AmqpsExceptionTranslator.convertToAmqpException(AmqpsExceptionTranslator.java:36)
 at com.microsoft.azure.sdk.iot.device.transport.amqps.AmqpsIotHubConnection.onTransportError(AmqpsIotHubConnection.java:761)
 at org.apache.qpid.proton.engine.BaseHandler.handle(BaseHandler.java:191)
 at org.apache.qpid.proton.engine.impl.EventImpl.dispatch(EventImpl.java:108)
 at org.apache.qpid.proton.reactor.impl.ReactorImpl.dispatch(ReactorImpl.java:324)
 at org.apache.qpid.proton.reactor.impl.ReactorImpl.process(ReactorImpl.java:291)
 at com.microsoft.azure.sdk.iot.device.transport.amqps.IotHubReactor.run(IotHubReactor.java:28)
 at com.microsoft.azure.sdk.iot.device.transport.amqps.AmqpsIotHubConnection$ReactorRunner.call(AmqpsIotHubConnection.java:824)
 at java.util.concurrent.FutureTask.run(FutureTask.java:266)
```

Solution

If this is the case, your network is blocking AMQPS traffic. For a rapid solution, you can switch to AMQPS over WebSocket by changing the setting in the **Cloud Connector→Connector Impl→Message Client→-Transport**. If you really want to use AMQPS, you can try working with your IT administration to modify the network settings to allow this protocol.

For most of the Honeywell networks, AMQPS is blocked by default. So, any device on the Honeywell corporate network (or our various internal test networks) will need to use AMQPS over WebSocket.

Chapter 8 Components and other references

Topics covered in this chapter

- Cloud Connector (cloudConnector-CloudConnector)
- Connector Impl (cloudSentienceConnector-SentienceConnectorImpl)
- Iot Hub Message Client (cloudIotHubDep-IotHubMessageClient)
- High/Low Priority Queues (cloudIotHubDep-MessageQueue)
- MessageClientUpgrader (cloudIotHubDep-MessageClientUpgrader)
- Cloud Event Recipient (nCloudDriver-CloudEventRecipient)
- Niagara Cloud Network (nCloudDriver-NiagaraCloudNetwork)
- Cloud Write Controller (nCloudDriver-CloudWriteController)
- Cloud Write Info (nCloudDriver-CloudWriteInfo)
- Cloud Alarm Recipient (nCloudDriver-CloudAlarmRecipient)
- Cloud Sentience Device (nCloudDriver-CloudSentienceDevice)
- Cloud Point Folder (nCloudDriver-CloudPointFolder)
- Cloud Point Device Ext (nCloudDriver-CloudPointDeviceExt)
- Cloud Proxy Ext (nCloudDriver-CloudProxyExt)
- Cloud Alarm Ext (nCloudDriver-CloudAlarmExt)
- Cloud History Device Ext (nCloudDriver-CloudHistoryDeviceExt)
- Cloud History Folder (nCloudDriver-CloudHistoryFolder)
- Cloud History Export (nCloudDriver-CloudHistoryExport)
- Cloud Commands Device Ext (nCloudDriver-CloudCommandsDeviceExt)
- Cloud Authentication Scheme (nCloudDriver-CloudAuthenticationScheme)
- Cloud Trust Manager (nCloudDriver-CloudTrustManager)
- Cert Trust Mapping (nCloudDriver-CertTrustMapping)
- Jwks Trust Mapping (nCloudDriver-JwksTrustMapping)
- Cloud User Mappings (nCloudDriver-CloudUserMappings)
- User Mapping (nCloudDriver-UserMapping)
- Role Mappings (nCloudDriver-RoleMappings)
- Cloud Device Manager (nCloudDriver-CloudDeviceManager)
- Cloud Point Manager (nCloudDriver-CloudPointManager)
- Cloud History Export Manager (nCloudDriver-CloudHistoryExportManager)
- Event messages and system commands

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**

Also discussed in this section are cloud-specific event messages and commands, as well as extensibility.

Cloud Connector (cloudConnector-CloudConnector)

This component enables a **NiagaraStation** running in a localhost or on an embedded controller to securely connect to a specific cloud platform.

For example, the **CloudConnector_Sentience nCloudDriver** component provides device registration and a secure connection to the Honeywell Sentience cloud platform for use with the Niagara CloudNiagara Cloud and other cloud services.

In addition to enabling cloud connectivity, the cloud connector authenticates the device to Niagara Cloud; maintains a message queue for sending messages to and from the cloud; raises an alarm if a connectivity problem occurs; and provides a bearer token to other services to identify those as coming from a properly registered device.

The cloud connector service generates alerts for the fault state, that is `Cannot connect to Cloud` and routes them to the alarm class specified in the service's **`Alarm Source Info`** property.

Device Registration is the default view that enables a user to register a device.

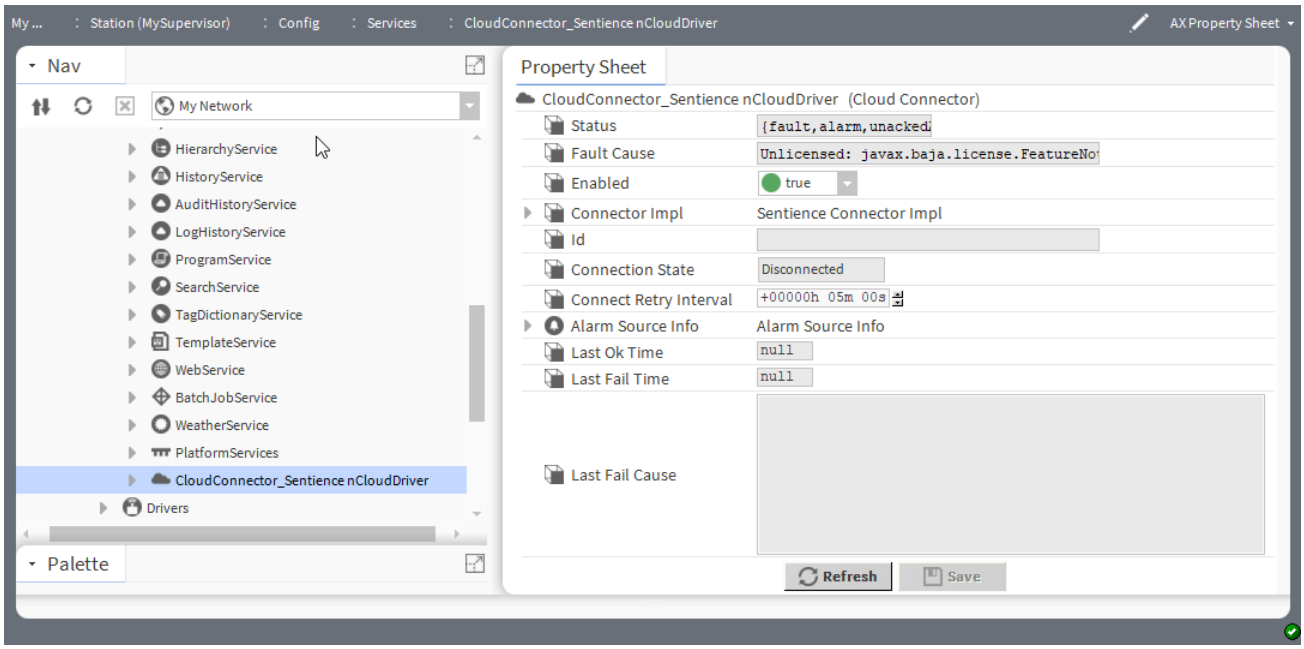
This component is found in the **`cloudSentienceConnector`** palette.
In the Niagara Cloud release 2018_4 and later, the connectors in the **`cloudSentienceConnector`** palette are renamed for clarity. The intended use for each one is as follows:

- **CloudConnector_Sentience nCloudDriver** supports the Niagara Cloud and .
- **CloudBackupService** supports the JACE-8000 and later controllers
- Use "CloudConnector_Sentience cloudBackup Only" for CloudBackupService when nCloudDriver is not supported (JACE-7 and older).
- Upgrade From cloudBackup Only to nCloudDriver supports installations that initially provided a connector that was for cloud backups only, or if you had an older connector that pre-dates the release of Niagara Cloud.

  Place **Upgrade From cloudBackup Only to nCloudDriver**under **CloudConnector/Connector Impl** and select the action **Switch To Iot Hub Message Client**. A JACE-8000 or above, as well as the **`CloudIotHubDep-rt.jar`** module are required for this upgrade.

For legacy platforms (JACE-3, JACE-6, and JACE-7) and for JACE-8000 platforms that are only used for cloud backups, only use the CloudConnector_Sentience cloudBackup Only component. This component does not support the Niagara Cloud, so this driver should not be used with legacy JACE platforms or with JACE-8000 platforms that are only used for cloud backups.

Figure 32 CloudConnector_Sentience nCloudDriver properties



To access these properties, expand **Config→Services**, scroll to **CloudConnector_Sentience nCloudDriver**, right-click the driver and click **Views→AX Property Sheet**.

Name	Value	Description
Status	read-only	Indicates the condition of the connector at the last check. • `{ok}` indicates that the cloud connector component is licensed, registered with Sentience and Niagara Cloud, and successfully connected. • `{down}` indicates that the cloud connector is not connected to the Niagara Cloud, perhaps it is not registered, or possibly loss of network connection. • `{disabled}` indicates that the Enable property is set to `false`. • `{fault}` indicates another problem. Check the Fault Cause property for more information.
Fault Cause	read-only	Indicates the reason why a system object (network, device, component, extension, etc.) is not working (in fault). This property is empty unless a fault exists.
Enabled	`true` (default)or `false`	Activates (`true`) and deactivates (`false`) use of the object (network, device, point, component, table, schedule, descriptor, etc.).
Connector Impl	additional properties	Contains read only properties that show registration and connection URLs, which determine how the device will be registered and connected. The topic **SentienceConnectorImpl** documents these properties.
Id	text string	Defines a globally unique identifier for each host, supplied by the Identity Provider with which the connector is registered. For the Honeywell Forge cloud platform, this is a system GUID (Globally Unique Identifier) that uniquely identifies the station within the cloud platform.
Connection State	read-only	Indicates whether or not the cloud connector is currently connected to the specified cloud platform. `Disconnected`, `Connected`, `Pending Connect`
Connect Retry Interval	hours minutes seconds (defaults to 20 seconds	Configures the amount of time between attempts to establish a connection upon a failure to connect.
Alarm Source Info	additional properties	Contains a set of properties for configuring and routing alarms when this component is the alarm source. For property descriptions, refer to the *Niagara Alarms Guide*
Last OK Time	read-only	Displays the last date and time the cloud connector was functioning as expected.
Last Fail Time	read-only	Displays the last date and time the cloud connector failed to connect.
Last Fail Cause	read-only	Displays the reason for the last failure of the cloud connector to connect.

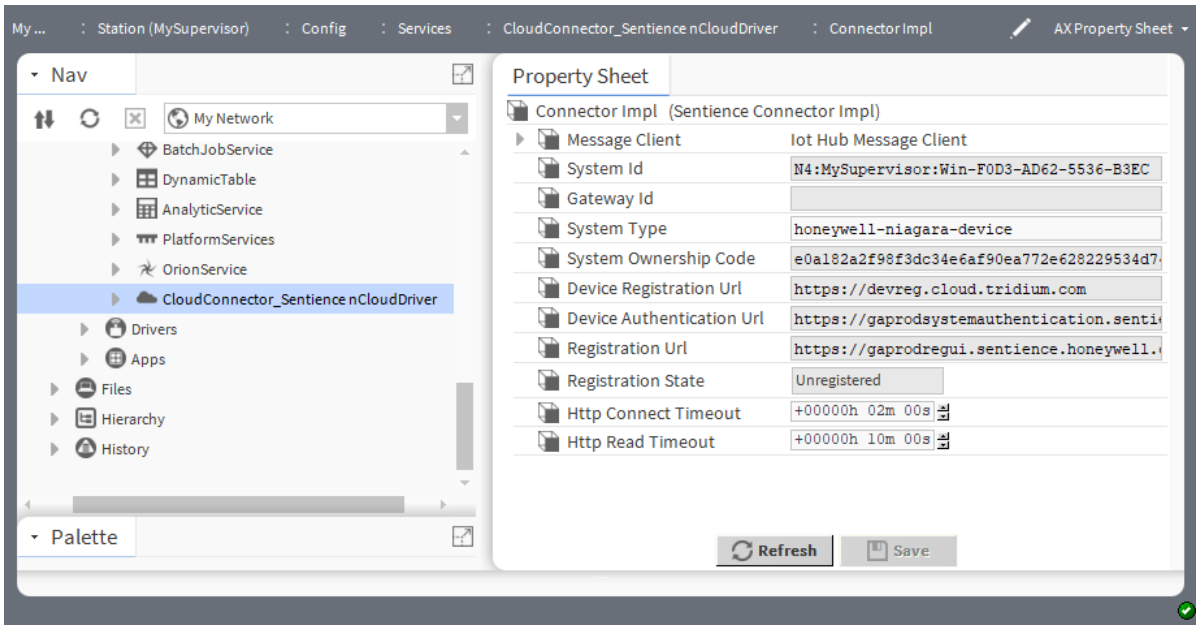
Actions

- **Reconnect** — This helps to reconnect the cloud.

Connector Impl (cloudSentienceConnector-SentienceConnectorImpl)

These properties are specific to **SentienceConnectorImpl**. Another ConnectorImpl, such as "AWSConnectorImpl", or "BluemixConnectorImpl", would not have these same properties.

Figure 33 Connector Impl properties



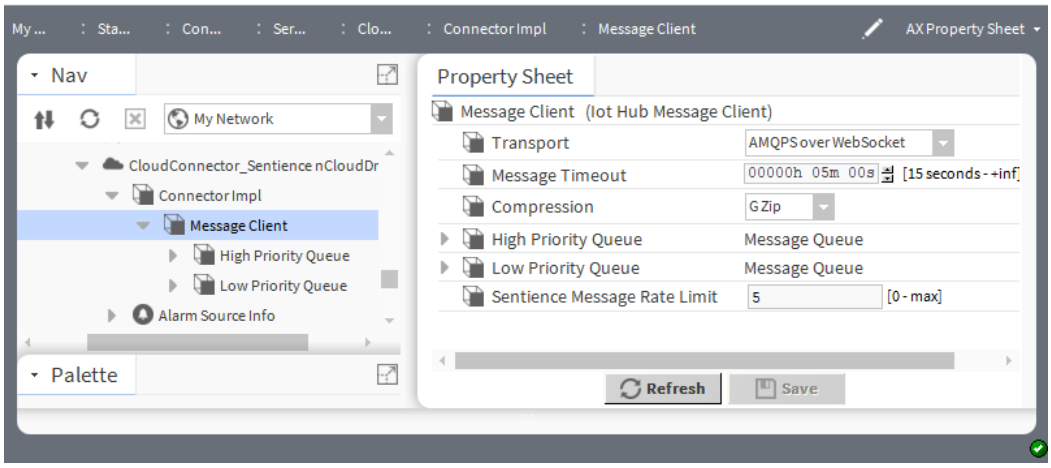
Name	Value	Description
Message Client	additional properties	Describes properties used to control messaging to the cloud platform. For legacy JACEs, this client is not used. It is not a NullMessageClient. For JACE-8000 and Supervisor platforms configured for Niagara Cloud, this is an IotHubMessageClient, which can be configured to optimize messaging to the IotHub. For JACE-8000 and Supervisor platforms, installing the cloudIotHubDep-rt module and the cloud connector_Sentience nCloudDriver connector ensures that the IotHubMessageClient is available.
System Id	read-only	Contains a logical identifier for the station outside of the provisioning lifecycle, and is used in registering the system with the cloud platform's identity provider.
System Type	text (defaults to blank)	The component found in the module palette. If you use a branded **`nCloudDriver`** palette that is appropriate to the NCHSD application, its cloud connector comes with a predefined **`System Type`** for the intended application.)Configures the identity of the station in the cloud outside of the provisioning lifecycle. Registration with the Niagara Cloud requires a value for this property. Sentience uses this property to group systems of similar origin and brand for data segregation purposes.

Name	Value	Description
		It also uses this property as a global access rights management mechanism. Applications and users must have rights to a particular **`System Type`** to be able to interact with systems registered to that **`System Type`**. For example, a **NiagaraStation** registered with **`System Type`** "brand A" cannot be accessed by a user or application whose permissions were for "brand B". There are additional permissions rules, but this global one requires that you pay close attention to getting the **`System Type`** right to ensure successful device registration.
System Ownership Code	read-only	Displays a unique code used to prove physical ownership and/or possession of the device in question.
Device Registration Url	read-only	Reports the endpoint for the Niagara Cloud web service that registers this device's identity with the cloud identity provider.
Device Authentication Url	read-only	Reports the endpoint for authenticating the device; that is., establishing that this device is the device whose identity was previously registered with the identity provider.
Registration Url	read-only	Reports the endpoint of the cloud platform identity provider for device registration.
Registration State	read-only	Reports the current state of device's registration with the identity provider: Unregistered, Registered, Needs Provisioning
Http Connect Timeout	hours minutes seconds (defaults to two minutes)	Configures the **CloudConnector** for the Niagara Cloud Honeywell Sentience Driver. This property is not for Backup as a Service.
Http Read Timeout	hours minutes seconds (defaults to 10 minutes)	Configures the **CloudConnector** for the Niagara Cloud Honeywell Sentience Driver. This property is not for Backup as a Service.

Iot Hub Message Client (cloudIotHubDep-IotHubMessageClient)

This component communicates with the Sentience IoT Hub using AMQPS (Advanced Message Queuing Protocol, *http://www.amqp.org/*). You can use its properties to tune performance.

Figure 34 Iot Hub Message Client properties



To access these properties, expand **Config→Services→CloutConnector_Sentience nCloudDriver→Connector impl**, right-click **Message Client** and click **Views→AX Property Sheet.**

Name	Value	Description
Transport	AMQPSoverWebSocket (default), AMQPS	Sends messages to the IotHub. The AMQPSoverWebSocket (AMQPS/WS) option connects to Honeywell Forge using the HTTPS web port, typically 443. The AMQPS transport option connects to Honeywell Forge on port 5671. **NOTE:** You need to configure firewalls to allow outbound connections from NiagaraStations to the hosts specified in the prerequisites for the procedure, “Setting up device internet access”, and for the port indicated by the selected Transport value (either AMQPSoverWebSocket or AMQPS).
Message Timeout	hours minutes seconds (defaults to five minutes)	Specifies how long to wait for a response for a message sent to the cloud platform. **CAUTION:** The Message Timeout value should remain at the default value. Adjust this value only upon recommendation by your Support channel.
Compression	drop-down list	Configures the driver to compress messages.
High Priority Queue	additional properties	The topic “High/Low Priority Queues” documents these properties.
Low Priority Queue	additional properties	The topic “High/Low Priority Queues” documents these properties.
Sentience Message Rate Limit	5 (default), 0 (no limit)	Configures the maximum number of messages/second from station to Honeywell Forge. Default is recommended for Honeywell Sentience starter environment instance. This property configures the CloudConnector for the Niagara Cloud. It does not support Backup as a Service.

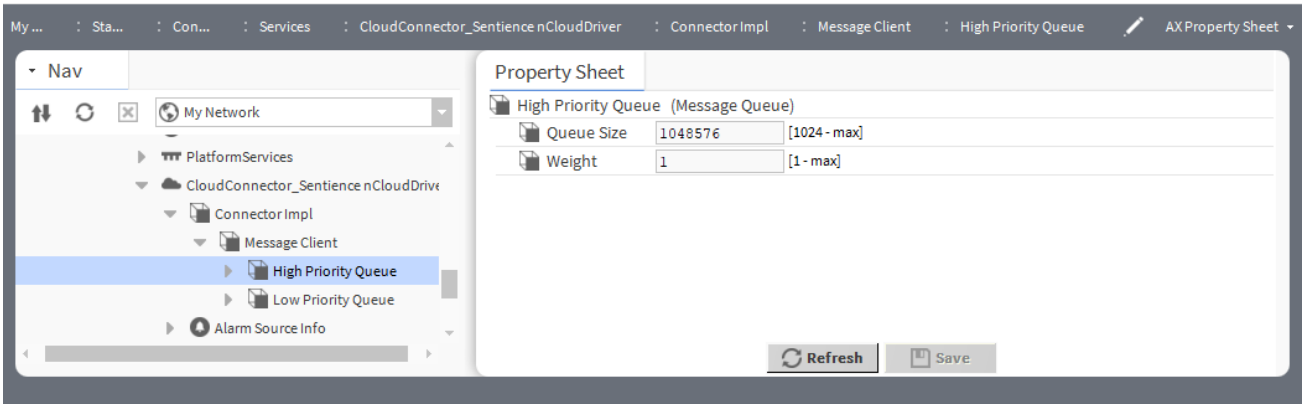
Actions

- **Reset Metrics** — this resets all the metrics.

High/Low Priority Queues (cloudIotHubDep-MessageQueue)

The driver sends data to the cloud by taking messages from the priority queues in a round robin fashion according to the **`Weight`** property value for each queue. For example, if the High Priority Queue has a weight of 2 and the Low Priority Queue has a weight of 1, assuming both queues have data to send, the IotHubMessageClient sends the two high priority messages first followed by the one low priority message.

Figure 35 High/Low Priority Queues properties



To access these properties, expand **Config→Services→CloudConnector_Sentience nCloudDriver→Connector Impl→Message Client**, double-click **High Priority Queue** or **Low Priority Queue**.

Name	Value	Description
Queue Size	number (defaults to 1048576	Defines the amount of data that can be placed in the queue before it will stop accepting new messages.
Weight	1 (default)	The weight property sets this queue's relative weight for message selection. A higher weight value will cause the queue to consume a larger share of the available bandwidth.

MessageClientUpgrader (cloudIotHubDep-MessageClientUpgrader)

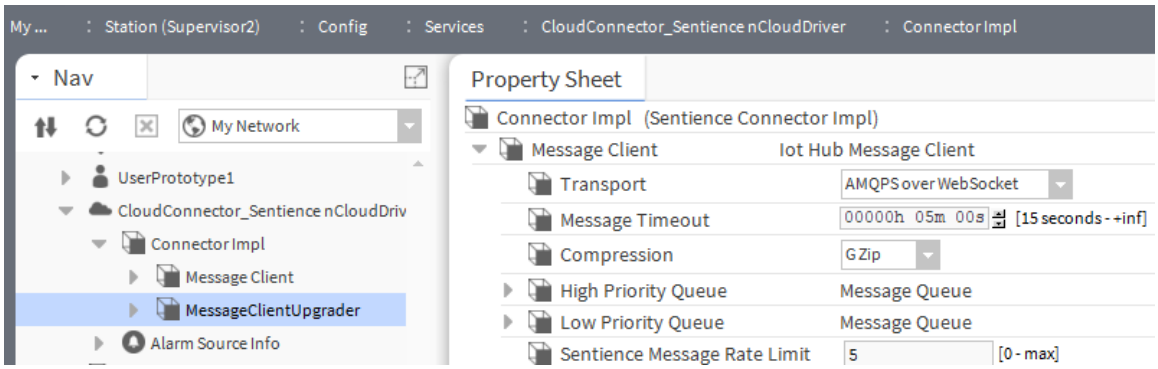
This component facilitates the upgrading of messages from the **`cloudBackup`** driver to the **`nCloudDriver`**. It has no properties.

This component is available in the **`cloudSentienceConnector`** palette.

This component makes a slight change to the installed cloud connector to use the modules for the Niagara Cloud.

To use this component, from the **`cloudSentienceConnector`** palette, expand **Upgrade From cloudBackup Only to nCloudDriver**, move **MessageClientUpgrader** to the **Connector Impl** folder, then right-click **MessageClientUpgrader** and select **Actions→Switch to IoT Hub Message Client**.

The **Message Client** changes from `Null Message Client` to `IoT Hub Message Client`.



Actions

- **Switch To Iot Hub Message Client** changes to the Iot Hub Message Client.

Cloud Event Recipient (nCloudDriver-CloudEventRecipient)

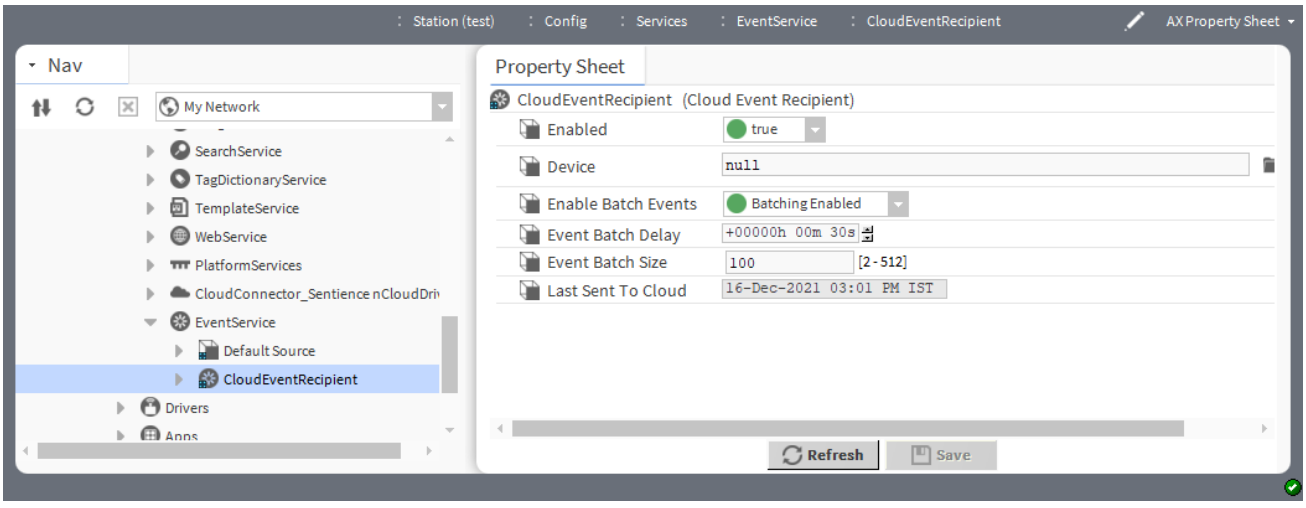
In Niagara 4.9 and later, the **CloudEventRecipient** is used much like the **CloudAlarmRecipient**, but with the Niagara EventService.

This generic component is configured to receive event data from Niagara as they are generated and send them to the cloud using the configured factory and the configured **CloudConnector**. The protocol used is determined by the configured **CloudSentienceDevice** protocol Type.

NOTE: A configured **NiagaraCloudNetwork** object must exist in the station in order for the **`CloudEvent-Recipient`** to function properly. Additionally, the **CloudConnector** must be properly configured in the station's Services.

The **`DefaultSource`** component is found in the **`event`** palette, while the **`CloudEventRecipient`** component is in the **`nCloudDriver`** palette.

Figure 36 CloudEventRecipient properties



To access these properties, expand **Config→Services→Event Service**, double-click **CloudEventRecipient**.

Property	Value	Description
Enabled	`true` (default) or `false`	Enables and disables the routing of events to the recipient. If it is set to `false`, trap events are not routed.
Device	`null` (default), `CloudSentienceDevice`	Indicates the CloudSentienceDevice (under NiagaraCloudNetwork) used for routing events.
Enable Batch Events	`Batching Enabled` or `Batching Disabled` (default)	Enables and disables batch events.
Event Batch Delay	`00000h 00m 30s` (default)	Specifies the amount of time to delay before sending batch events to the cloud.
Event Batch Size	`100` (default)	Specifies the size (number of events). Range is from 2 to 512.
Last Sent To Cloud	`null` (default)	Timestamp shows the last time events batch was sent to the cloud.

Action

- **Process** — A driver triggers the event on the **EventService** providing the timestamp, source, etc., for the event which is sent to the cloud.

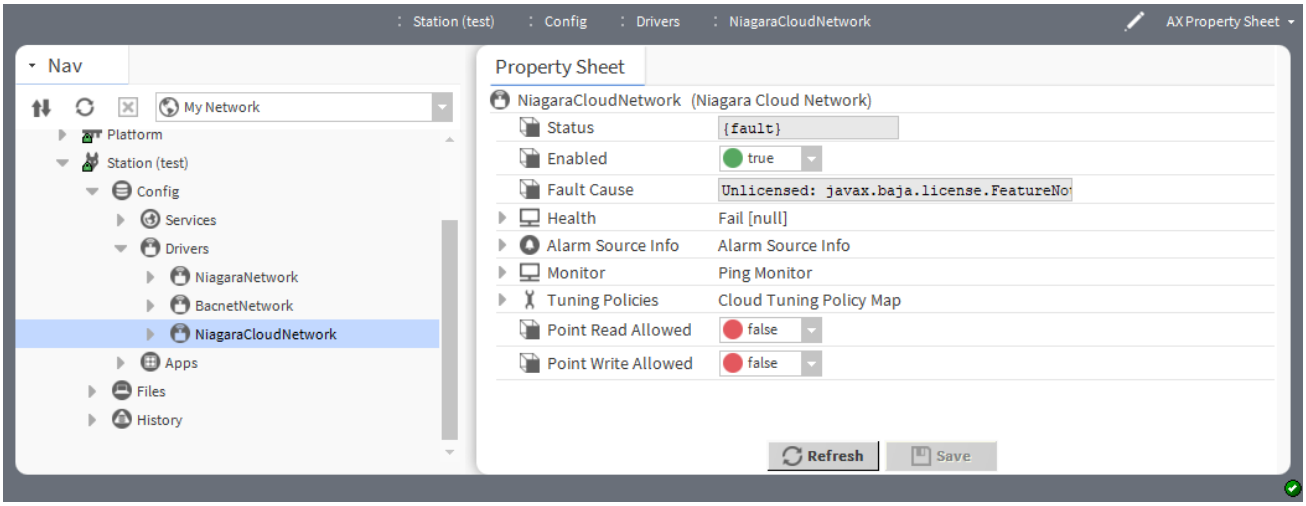
Niagara Cloud Network (nCloudDriver-NiagaraCloudNetwork)

This component is a container for most components required for communicating with the cloud. This component provides logical organization for many types of cloud platforms.

The **NiagaraCloudNetwork** structure, shown here, adheres to the standard driver framework utilities built into Workbench. The overall intent is to provide generic Niagara components that can be used to accomplish specific goals without having to program in cloud platform provider-specific terms. Such specifics are held within the CloudDevice in what is known as a "concrete cloudFactory" and are used by the child device extensions as needed.

This component is available in the **`nCloudDriver`** palette. The **Cloud Device Manager** is the default view of this component.

Figure 37 Niagara Cloud Network properties



To access these properties, expand **Config→Drivers**, right-click **NiagaraCloudNetwork** and click **Views→Ax Property Sheet**.

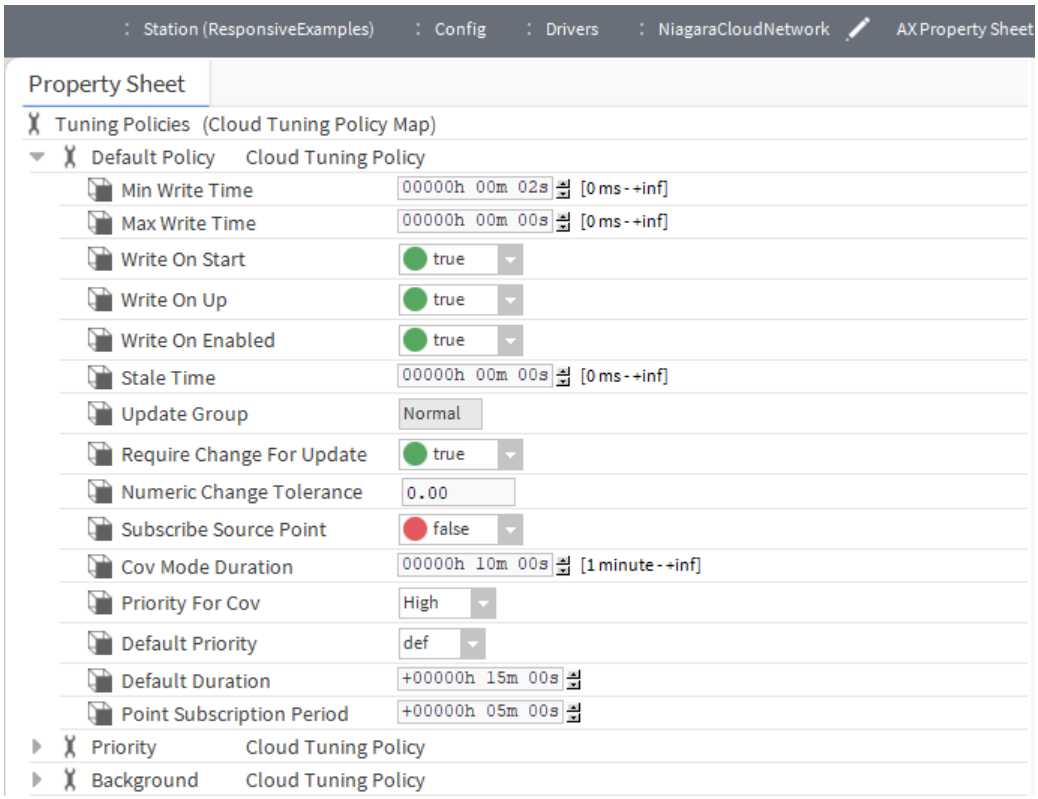
In addition to the standard properties (Status, Enabled, Fault Cause, Health and Alarm Source Info), these properties are unique to this component.

Name	Value	Description
Monitor	additional properties	Configures a network's ping mechanism, which verifies network health. This includes verifying the health of all connected objects (typically, devices) by pinging each device at a repeated interval. The *Niagara Drivers Guide* documents these properties.
Tuning Policies	additional properties	Holds the tuning policies governing the behavior of cloud proxy points with respect to communication to the cloud. For details see, Tuning Policies properties, page 88.
Point Read Allowed	`true`, `false` (default)	Must be set to "true" if point read commands are sent from the cloud.
Point Write Allowed	`true`, `false` (default)	Must be set to "true" if point write commands are sent from the cloud.

Tuning Policies properties

NOTE: The Point Subscription Period property, described below, is available starting in Niagara 4.10u2 and later versions.

Figure 38 Tuning Policies Property Sheet view



In addition to the common properties (Min Write Time, Max Write Time, Stale Time) the following properties are available.

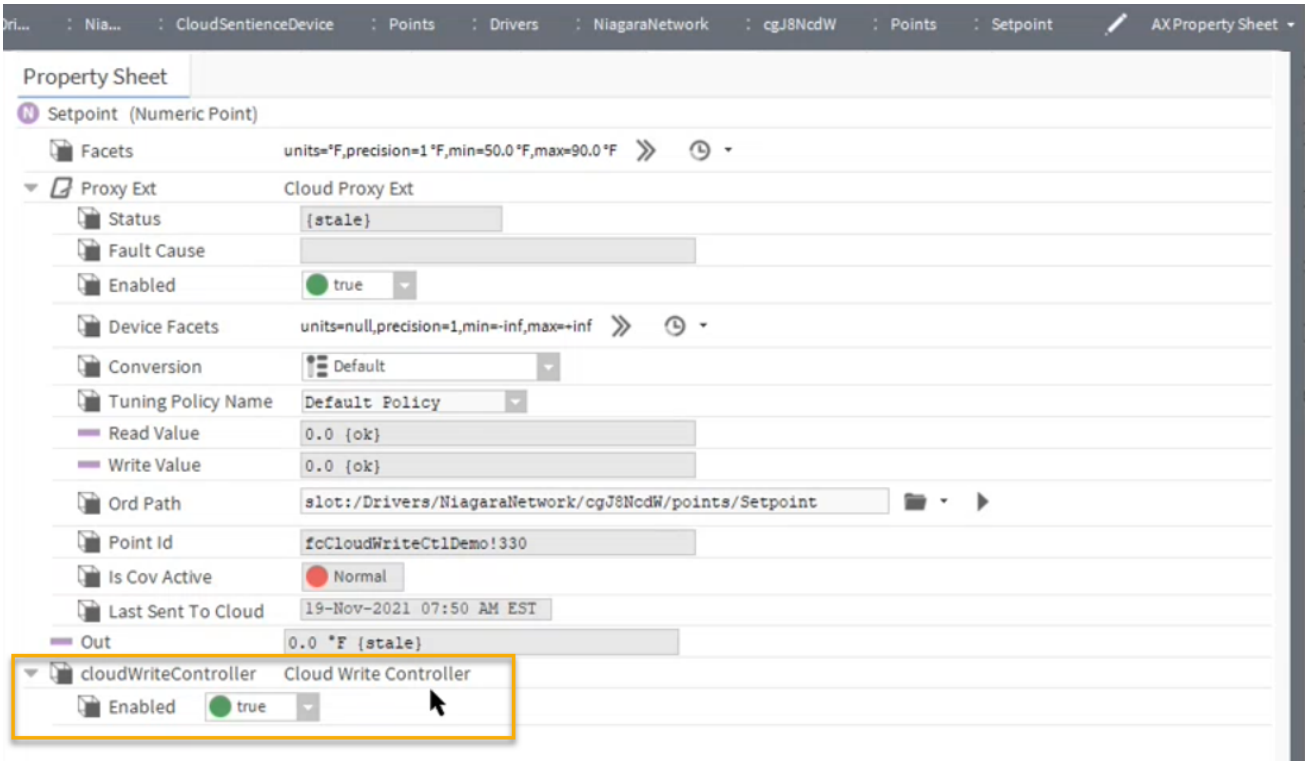
Name	Value	Description
Write On Start	true (default), false	Determines behavior of writable proxy points at station start-up. This property applies to the Default Policy, Priority and Background tuning policies. If “true”, (default) a write occurs when the station first reaches steady state. If set to “false”, a write does not occur when the station reaches steady state. Consider setting this to false in most tuning policies, except for tuning policies selectively assigned to more critical writable proxy points. This is particularly important for large networks with many writable proxy points.
Write On Up	true (default), false	Determines behavior when writable proxy point (and parent device) transitions from “down” to “up.” This property applies to the Default Policy, Priority and Background tuning policies. If “true”, (default) a write occurs when the parent device transitions from down to up. If set to “false”, a write does not occur when the parent device transitions from down to up
Write On Enabled	true (default), false	Determines behavior when a writable proxy point’s status transitions from disabled to normal (enabled). This property applies to the Default Policy, Priority and Background tuning policies. If “true”, (default) a write occurs when writable point transitions from disabled. If set to “false”, a write does not occur when writable point transitions from disabled. The disabled-to-enabled status transition can be inherited globally by points if the parent device had been set to disabled—or network-wide if the driver network was set to disabled. Therefore, be aware that if left at true in tuning policies, that all associated writable points receive a write upon either the device or network when it transitions from status disabled to enabled.
Update Group	`fast`, `normal` (default), `slow`	Specifies the update frequency for batch point updates. • For the Default Tuning Policy, specifies the period for normal (standard) batch point updates. Default setting is 5 minutes. Minimum limit is 1 minute. • For the Priority Tuning Policy, specifies the period for priority batch point updates. The default setting is 30 seconds. Minimum limit is 5 seconds. • For the Background Tuning Policy, specifies the period for background batch point updates. The default setting is 15 minutes. Minimum limit is 5 minutes.
Require Change For Update	`true`, `false`	Specifies whether this point must have a value change to be included in batch update.
Numeric Change Tolerance	numeric	Specifies the minimum change that will cause a numeric point to be included in batch update, if the Require Change For

Name	Value	Description
		Update property is "true". Boolean, String, Enum points do not have a change tolerance; any change drives an update. For any point type, including Numeric Points, any change of point status will add the point to the batch update regardless of the point value.
Subscribe Source Point	`true`, `false` (default)	Flag for controlling whether the source point is forced into subscription by the CloudProxyPoint. If set to "false" (the default value) it will not affect whether the source point is subscribed. When set to "true" it causes the CloudProxyPoint to force a permanent subscription of the source point. **NOTE:** Setting this property to "true" may affect the performance of the driver network containing the source point.
COV Mode Duration	`+00000h 10m 00s` (default)	Specifies how long COV points will stay in COV before exiting COV mode.
Priority For COV	`High` (default), `Low`	Specifies whether COV messages will be sent with high or low priority.
Default Priority	`none`, `in1` through `in16`, `fallback`	Starting in Niagara 4.9 (NCHSD 2019.2), this property is used to configure the priority for writes incoming from the cloud, when using System Command writes, and the Priority parameter is not present.
Default Duration	`true`, `false` (default)	Starting in Niagara 4.9 (NCHSD 2019.2), this property is used to configure writes incoming from the cloud, when using System Command writes and the **`Duration`** parameter is not present.
Point Subscription Period	`+00000h 5m 00s` (default)	If the **`Subscribe Source Point`** property is set to `false`, then on live value read using Honeywell Cloud Sentience APIs, respective source points are added to the Niagara subscription (if not already subscribed) for this specified period of time. Once the time expires, the points are automatically unsubscribed. **NOTE:** The nCloudDriver adds the points to Niagara subscription on live value read. Based on the tuning policy configuration of underlying drivers, the polling or COV mechanism is used to get the value from the remote devices. If the subscription time is set to +0000h 0m 00s, the points will not be subscribed through the nCloudDriver on live value read using Honeywell Cloud Sentience APIs.

Cloud Write Controller (nCloudDriver-CloudWriteController)

This component allows you to write values from the cloud to points of a remote device through the Forge Connect Gateway. When you add a point to the CloudSentienceDevice in your cloud network, the proxy extension automatically creates a cloudWriteController, which you can disable or leave enabled to allow write action.

Figure 39 cloudWriteController property



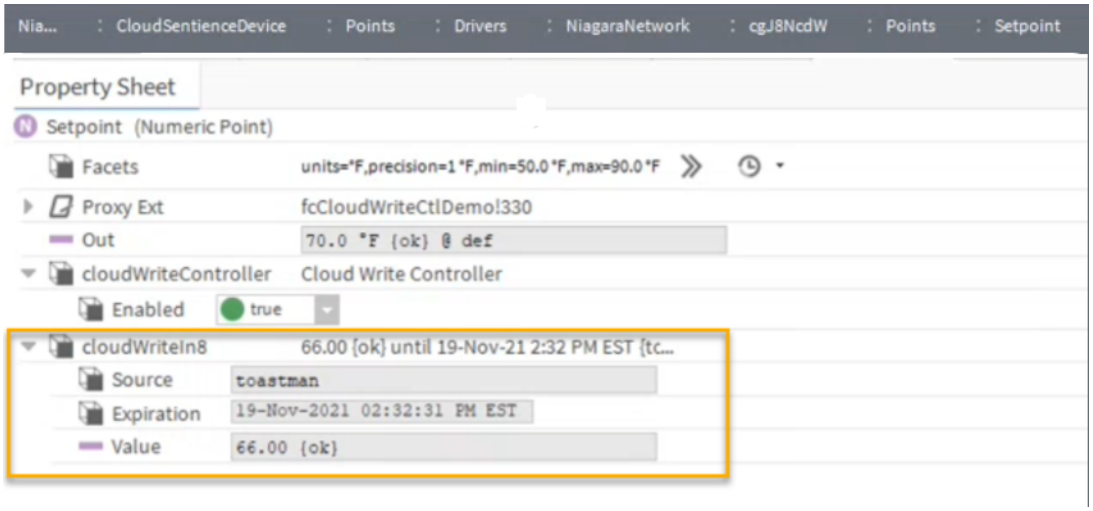
To access this property, expand **Config→Drivers→NiagaraCloudNetwork→Points** and expand **`cloudWriteController`**.

Property	Value	Description
Enabled	`true` (default), `false`	Activates (true) and deactivates (false) the write action from the cloud to the point in the remote device.

Cloud Write Info (nCloudDriver-CloudWriteInfo)

This component appears under the cloud proxy point in the NiagaraCloudNetwork when the cloud point receives a write action from the Forge Connect application.

Figure 40 cloudWriteInfo property



To access this property, expand **Config→Drivers→NiagaraCloudNetwork→Points** and expand **`cloudWriteInfo`**.

Property	Value	Description
Source	read-only	Displays the cloud source of the write action as provided in the system command source property, or if that is blank, the application ID of the sending client. For the case that the cloud driver harmonizes the value with the remote device due to a remote point update, the driver may update the field.
Expiration	read-only	Reports the date and time when the write action expires.
Value	read-only	Specifies the value of the write action received from the cloud.

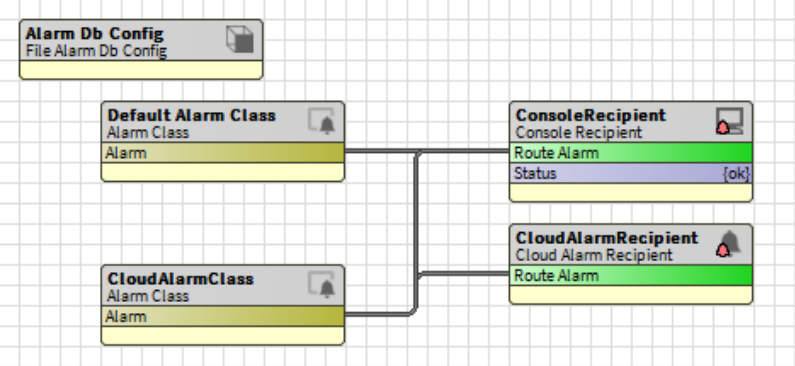
Cloud Alarm Recipient (nCloudDriver-CloudAlarmRecipient)

This component extends the standard AlarmRecipient. It functions the same as the standard AlarmRecipient component does except that the **CloudAlarmRecipient** component can be used to send alarms to a single configured cloud platform.

This generic component is configured to receive alarms from Niagara as they are generated and send them to the cloud using the configured factory and the configured CloudConnector. The protocol used is determined by the configured CloudSentienceDevice protocolType.

In order to use the **CloudAlarmRecipient**, you must place the component under the **Station→AlarmService** in the wire sheet view, as in the example shown.

Figure 41 CloudAlarmRecipient added to AlarmService Wire Sheet

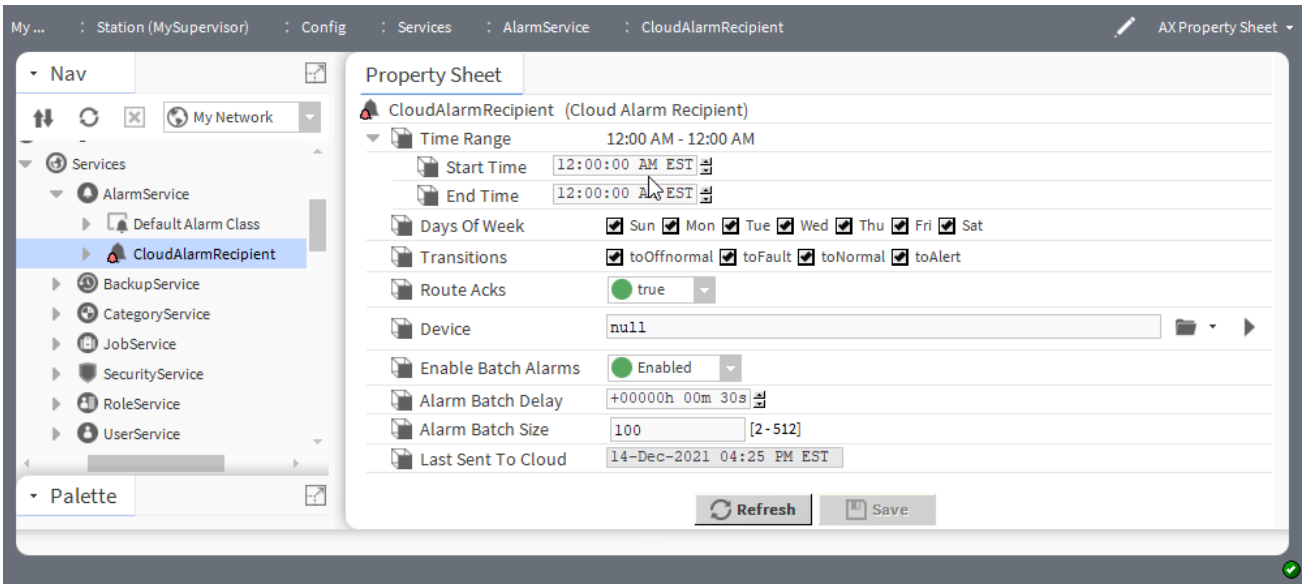


NOTE: A configured **NiagaraCloudNetwork** object must exist in the station in order for the**CloudAlarmRecipient** to function properly.

Notice that the **Alarm** slot of the Alarm Class component is linked to the **Route Alarm** slot of the **CloudAlarmRecipient** component. This ensures that alarms are routed to the **CloudAlarmRecipient** appropriately.

This component is available in the **`nCloudDriver`** palette.

Figure 42 Cloud Alarm Recipient properties



The **`Time Range`**, **`Days of Week`**, and **`Transitions`** properties are standard to all Alarm Recipients and detailed in the *Niagara Drivers Guide*. The **CloudAlarmRecipient**-specific configuration properties are described in the following table.

To access these properties, expand **Config→Services→Alarm Service**, double-click **CloudAlarmRecipient**.

Name	Value	Description
Time Range, Start Time	hours:minutes: seconds	Specifies when to begin reporting alarms.
Time Range, End Time	hours:minutes: seconds	Specifies when to end reporting alarms.
Days of Week	check boxes	Limits alarm reporting to a specific day or selected days of the week.

Name	Value	Description
Transitions	check boxes	Selects which alarm conditions to report.
Route Acks	`true` (default) or `false`	Enables (`true`) and disables (`false`) the routing of alarm acknowledgements to the recipient. The framework does not route trap (event notification) acknowledgements if you select `false`.
Device	`null` (default), `CloudSentienceDevice`	Indicates the CloudSentienceDevice to be monitored for alarms.
Enable Batch Alarms	`Enabled` (default), `Disabled`	Enables and disables the
Alarm Batch Delay	`00000h 00m 30s` (default)	Specifies the amount of time to delay before sending batch alarms to the cloud.
Alarm Batch Size	`100` (default)	Specifies the size Range is from 2 to 512.
Last Sent To Cloud	`null` (default)	Shows the last time alarms batch was sent to the cloud

Action

- **Route Alarm**: routes the alarm for selected alarm class.

Alarm Class considerations — Priority

In the Niagara Alarm Class, you can select a number from 0-255 to set the priority for **`toOffnormal`**, **`toFault`**, **`toNormal`** and **`toAlert`**. However, Honeywell Forge cloud platform has a different priority scheme. The following table maps a specific range of Niagara priority numbers to each Honeywell Forge priority.

Niagara priority number range	Honeywell Forge priority
0 - 31	Emergency – 1
32 - 64	Alert – 2
65 - 97	Critical – 3
98 - 130	Error – 4
131 – 163	Warning – 5
164 – 196	Notice – 6
197 - 229	Information – 7
230 – 254	Diagnostic – 8
255	Undefined – 0

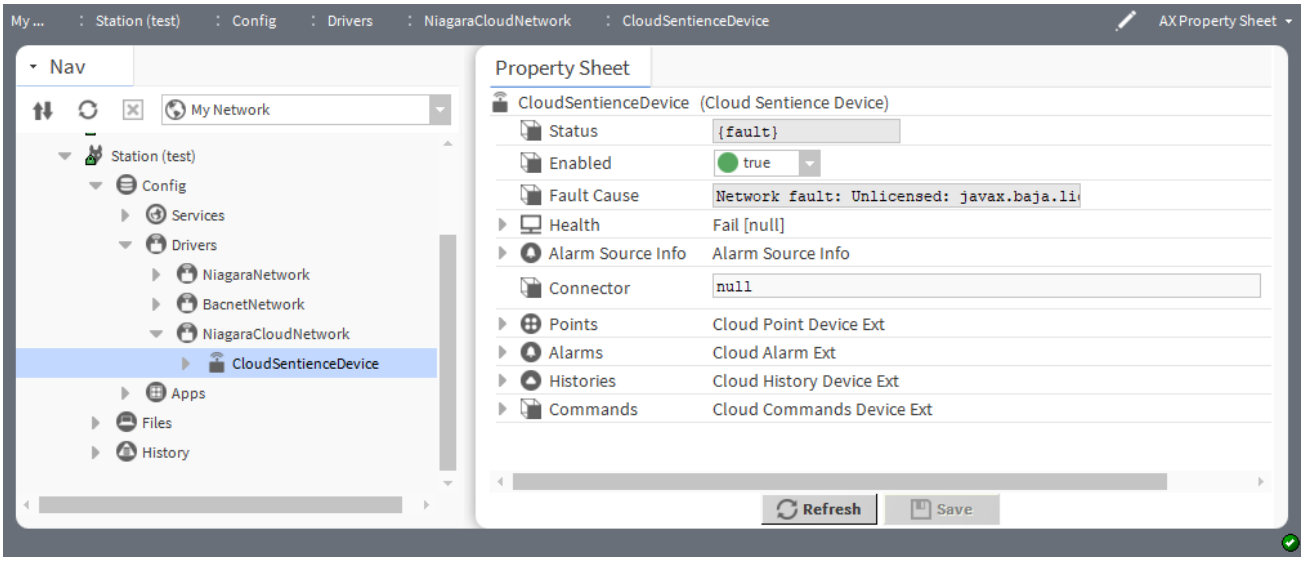
Cloud Sentience Device (nCloudDriver-CloudSentienceDevice)

The cornerstone of the NiagaraCloudNetwork is the CloudSentienceDevice which holds cloud-specific logic regarding which points to send, which histories to send, which CloudConnector to use when sending, as well as how to read messages sent back from the cloud.

The CloudSentienceDevice is configured for the CloudConnector_Sentience which contains the appropriate protocol for connecting to and communicating with the Honeywell Forge cloud platform. Within the CloudSentienceDevice is a specific type of "concrete cloudFactory" which is used to create the messages that this CloudDevice can send. These messages are specifically formatted to work with the cloud platform.

This component is available in the **`nCloudDriver`** palette.

Figure 43 Cloud Sentience Device Property Sheet



To access these properties, expand **Config→Drivers→NiagaraCloudNetwork**, double-click **CloudSentienceDevice**.

In addition to the standard properties (Status, Enabled, Fault Cause, Health and Alarm Source Info), these properties support this component.

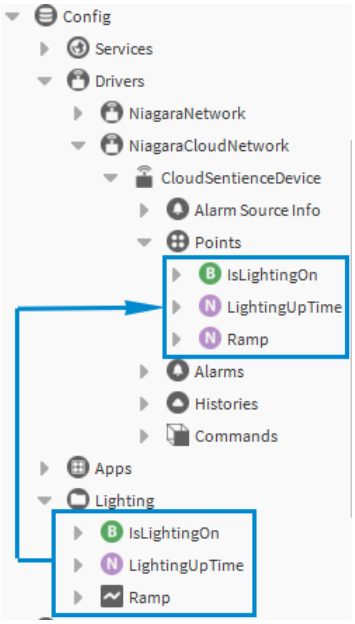
Property	Value	Description
Points	additional properties	Serves as the parent container for real-time values originating from the station. It also contains a set properties for exporting data to the cloud platform.
Alarms	additional properties	Handles the acknowledgement of alarms from the cloud.
Histories	additional properties	Contains a set of properties for logging historical data for this device
Commands	additional properties	Contains a set of registered command(s) for handling incoming messages.
Connector	null (defult), Ord path	Specifies the CloudConnectorService for this device.

Cloud Point Folder (nCloudDriver-CloudPointFolder)

This component is provided for organizing points, if desired. The standard control point types are in this folder as well.

The Cloud Point Manager is the default view for this component. Use this view to discover points inside the station and add them to the database of proxy points under the device, as shown.

Figure 44 Adding discovered station points to the database of proxy points

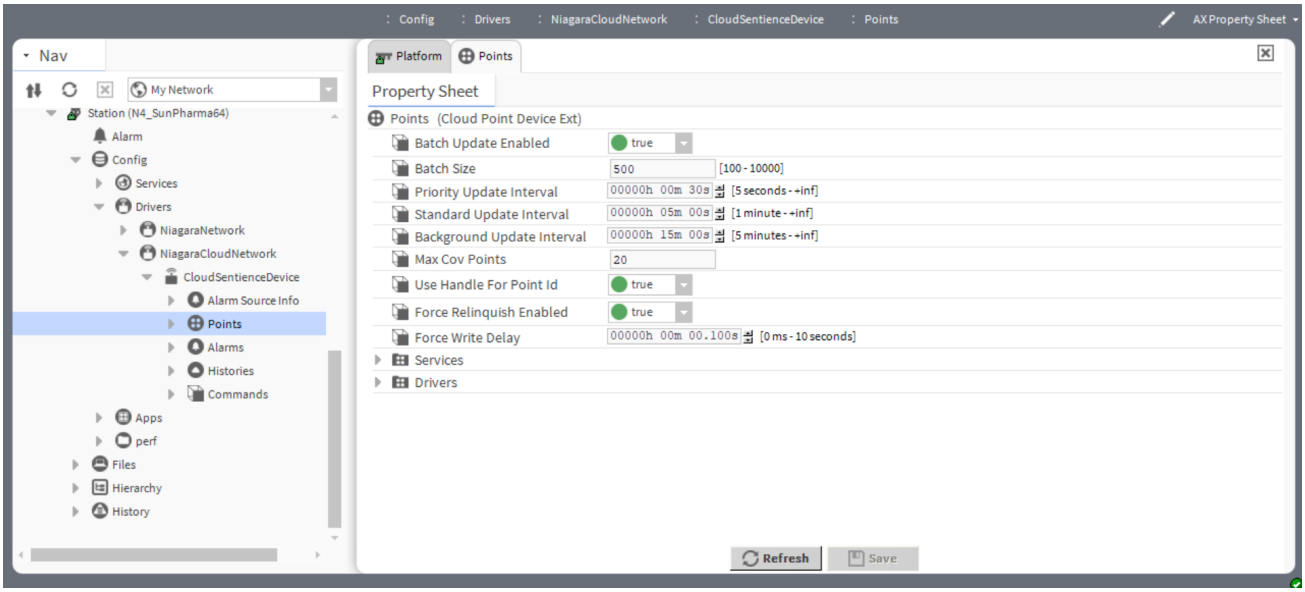


This component is available in the **nCloudDriver** palette.

Cloud Point Device Ext (nCloudDriver-CloudPointDeviceExt)

This component is the parent container for cloud proxy points that export station data to the cloud. All four base point types can be handled. Cloud proxy points are referenced and linked to station control points during discovery. Point data is written to the cloud platform either on change of value, or periodically using a batch update.

Figure 45 CloudPointDeviceExt properties



To access these properties, expand **Config→Drivers→NiagaraCloudNetwork→CloudSentienceDevice**, right-click **Points** and click **Views→AX Property Sheet**.

Name	Value	Description
Batch Update Enabled	`true` (default), `false`	Enables (true) and disables (false) batch updates. Setting this value to `false` can be useful to temporarily stop all traffic related to points for the purpose of diagnosing a problem with another aspect of the station.
Batch Size	`500` (default)	Specifies the number of point updates to be included in a single message. This may be tuned upward for larger stations to optimize bandwidth usage. Settings beyond around 5000 may not produce significant improvements due to restrictions on maximum message size by the SentienceIoTHub. This value is dependent upon the exact station structure, as the **`Point Id`** is included in the update. Its size depends upon the station folder structure. Stations with a flatter folder structure have shorter **`Point Id`**s, allowing the update message to fit more points. The tradeoff to this is that the flatter folder structure means more points are in each folder. Loading and viewing these folders might take longer. Stations with a deeper folder structure (or particularly long folder names) can fit fewer points in the update message (compression will mitigate this), but may have faster load times for viewing a particular folder.
Priority Update Interval	`0000h 00m 30s` (default)	Specified period for priority batch point updates as determined by the NiagaraCloudNetwork Priority Tuning Policy. Minimum limit is 5 seconds.
Standard Update Interval	`0000h 05m 00s` (default)	Specified period for normal batch point updates as determined by the NiagaraCloudNetwork Default Tuning Policy. Minimum limit is 1 minute.
Background Update Interval	`0000h 15m 00s` (default)	Specified period for background batch point updates as determined by the NiagaraCloudNetwork Background Tuning Policy. Minimum limit is 5 minutes.
Max Cov Points	`20` (default)	Specifies the maximum number of points that can be configured for COV messaging. **CAUTION:** This value should only be increased with care, as sending individual point updates is very inefficient and can cause the cloud platform provider to block systems communication to the cloud if too much traffic is sent.
Use Handle For Point ID	`true` (default), `false`	Configures the Point Id for the proxy points. It specifies whether Cloud Proxy Points will use Niagara Handle (true) or Slot Path (false) ord for generating their point ids. The option to select false is available only for backward compatibility, to allow installations to retain continuity of historical data that may have been pushed to the data storage using the old slot-path-based point id. The option to use slot path may be removed in a future release, so users are strongly encouraged to use this as a temporary option only. **NOTE:** When this property is changed, the Point Id of all proxy points is updated to reflect the new setting.

Name	Value	Description
Force Relinquish Enabled	`false` (default), `true`	As of Niagara 4.10u3 and later: `false` allows Forge Connect Niagara to change a point value at a certain priority level only if Forge Connect Niagara executed the current override. `true` allows Forge Connect Niagara to change a point value at a certain priority level only if Forge Connect Niagara did **not** execute the current override. For example, when this property is set to `true` and a local operator directly overrides a point value in a Supervisor station without using Forge Connect Niagara, another operator can use a cloud application (such as Centralized Control) and force a relinquish action that allows the cloud application to override the local change. **CAUTION:** When **`Force Relinquish Enabled`** is `true` and a cloud application initiates a relinquish of a point from the Cloud, Forge Connect Niagara triggers a write of the latest value to the specified priority, then, after a delay, relinquishes control of the point. The **`Force Write Delay`** property configures this delay. **Limitation**: Only the Niagara driver supports this property. It is not supported for any other driver. Keep this property set to `false` for points under all other drivers.
Force Write Delay	`00000h 00m 00.100s` (default)	As of Niagara 4.10u3 and later, specifies how long of a delay to allow between a point write and relinquish when **`Force Relinquish Enabled`** is set to `true`. You should base this time on the maximum time it takes for a point write from Niagara's Forge Connect. You may need to adjust this value based on the performance of each deployment. The qualified value for the Niagara driver (ndriver) in a lab environment is `00000h 00m 00.100s` (default value).

Actions

Learn Points — creates a **CloudPointFolder** for each folder in the station that it finds. This helps to organize the potentially large number of points into some logical structure. It also improves the performance for large stations significantly over creating all the cloud proxy points in the same folder. This is strongly recommended over standard point discovery and adding, for any station with more than several thousand points.

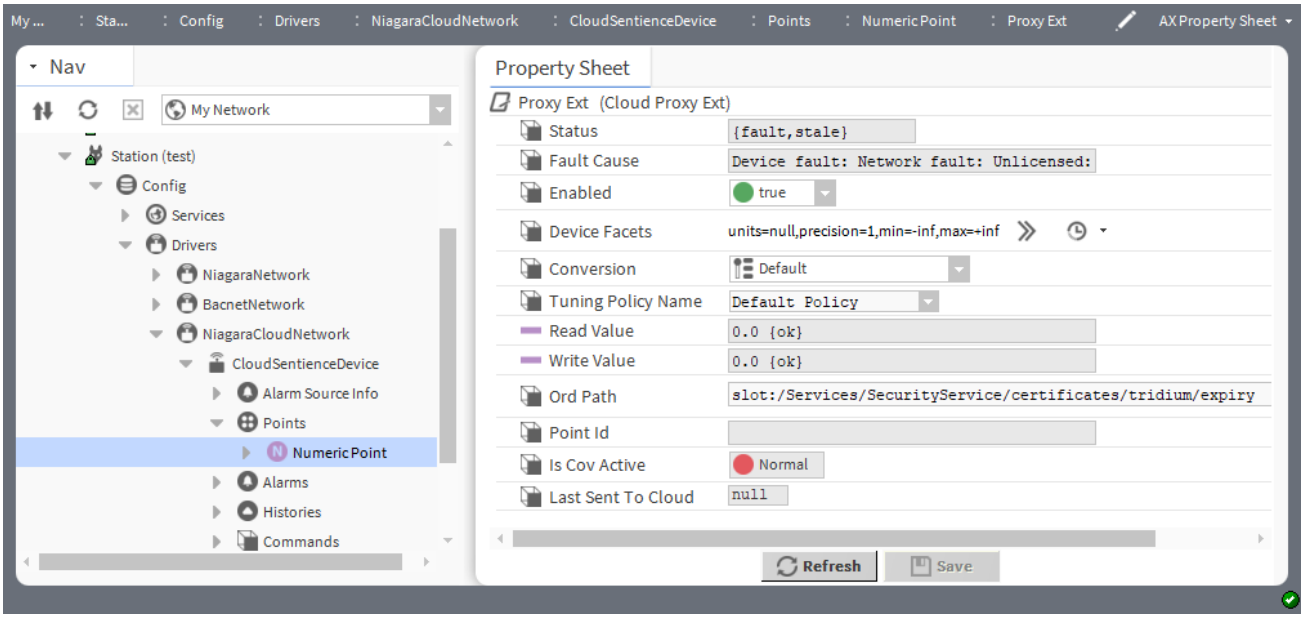
Cloud Proxy Ext (nCloudDriver-CloudProxyExt)

This component is a Niagara Proxy Extension configured to push point data to the cloud. Its configuration indicates from where the point's data originate, including details specific to the NiagaraCloudNetwork and Niagara Cloud.

The proxy points can be individually enabled/disabled. But they start out enabled by default, no manual step is required. Also, the point data is pushed to the cloud only on every change if it is configured for COV (Change Of Value).

NOTE: Only a small number of points should be configured for COV as it can cause significant performance problems. A better practice is to use batch updates.

Figure 46 Cloud Proxy Ext properties



To access these properties, expand **Config→Drivers→NiagaraCloudNetwork→CloudSentienceDevice→-Points→Numeric Point**, right-click **proxy Point** and click **Views→AX Property Sheet**.

In addition to the standard properties (Status, Fault Cause and Enabled), these properties support this component.

Name	Value	Description
Device Facets	trueText (default) or falseText	Facets contain key/value pairs applied to input and output values. These sample facets are for a boolean point. They will be different for other point types. • *trueText* is the text to display when output is true • *falseText* is the text to display when output is false. For example, you might want to set the key trueText to display "ON" and the key falseText to display "OFF." Other keys are units, precision, etc. You can edit Facets from a component Property sheet by clicking the >> icon to display the **Config Facets** window.
Conversion	drop-down list	Defines how the system converts proxy extension units to parent point units. `Default` automatically converts similar units (such as Fahrenheit to Celsius) within the proxy point. **NOTE:** In most cases, the standard `Default` conversion is best. `Linear` applies to voltage input, resistive input and voltage output writable points. Works with linear-acting devices. You use the Scale and Offset properties to convert the output value to a unit other than that defined by device facets. `Linear With Unit` is an extension to the existing linear conversion property. This specifies whether the unit conversion should occur on "Device Value" or "Proxy Value". The new

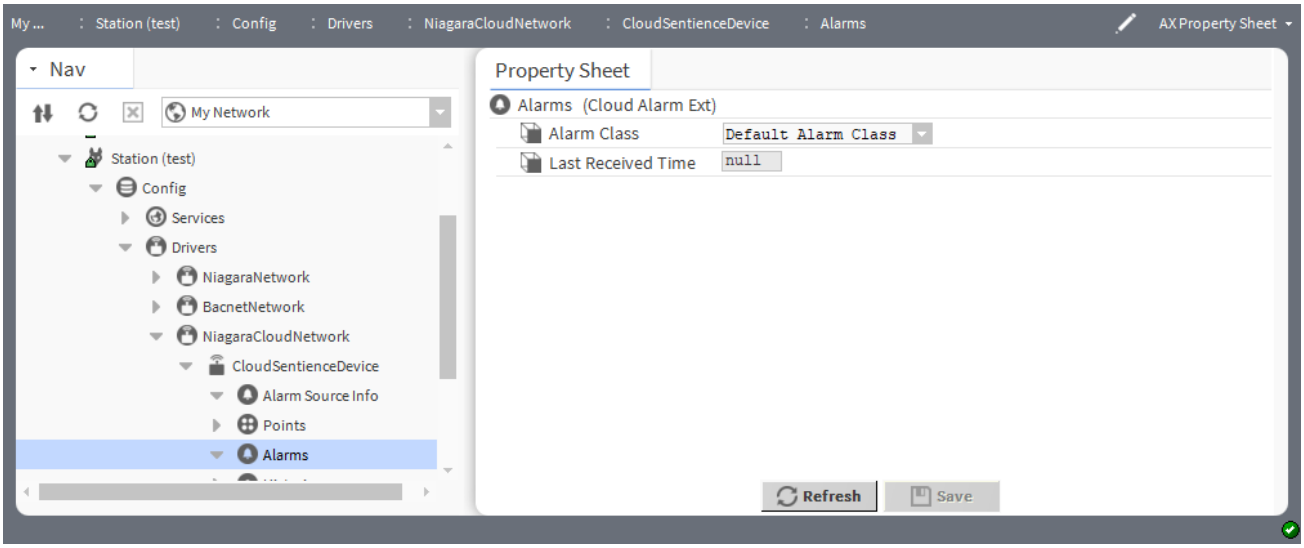
Name	Value	Description
		linear with unit convertor, will have a property to indicate whether the unit conversion should take place before or after the scale/offset conversion. `Reverse Polarity` applies only to Boolean input and relay output writable points. Reverses the logic of the hardware binary input or output. `500 Ohm Shunt` applies to voltage input points only. It reads a 4-to-20mA sensor, where the Ui input requires a 500 ohm resistor wired across (shunting) the input terminals. `Tabular Thermistor` applies to only a Thermistor input point and involves a custom resistance-to-temperature value response curve for Type 3 Thermistor temperature sensors. `Thermistor Type 3` applies to an Thermistor Input point, where this selection provides a “built-in” input resistance-to-temperature value response curve for Type 3 Thermistor temperature sensors. `Generic Tabular` applies to non-linear support for devices other than for thermistor temperature sensors with units in temperature. Generic Tabular uses a lookup table method similar to the “Thermistor Tabular” conversion, but without predefined output units.
Tuning Policy Name	Priority Policy, Default Policy, Background Policy	Defines rules for managing how station data are sent to and received from the cloud, including update group, COV, change requirements, and duration of COV mode and cloud writes.
Read Value	read-only	Shows the last value that was successfully sent to the cloud, for example "`71 {ok}`" or "`false {ok}`". Read Value is updated automatically upon successful write, as the value is not "read" from the cloud.
Write Value	read-only	Shows the last value written to the cloud, as well as the state and priority level, for example, “`70 {overridden} @8`”. The Write Value is a value that this station's source point has taken on, which needs to be written up to the cloud database.
Ord Path	slot:/	Defines the Ord path for the source point in the local station.
Point Id	read-only	Reports a unique ID (status/ord path) with the station name added to the point path. This is the reference value that is used to retrieve the data from the cloud, and for sending data from the cloud to the station.
Cloud Write Info	read-only	In Niagara 4.10/FC 1.2 and later, if a write is active each Cloud Write Info command coming in from the cloud receives its own slot on the cloud proxy extension dynamically, and links the value to the source station point at the priority level specified by the write. Cloud Write Info provides the following details: the Honeywell Forge Point ID of the point being written, the new value, the priority level at which to write, the duration that this write should remain active (after this time the write is cancelled), and the application or user issuing the write.
Is COV Active	read-only	Indicates if the Cloud Proxy Ext is currently in COV mode or not.

Name	Value	Description
		If a point is placed into COV, there is a property called **`Cov To Cloud Value`** that is added while the point is in COV mode, which contains the value from the source point, to be pushed to the cloud via COV update message.
Cov To Cloud Value	read-only	Not present by default, but if a point is placed into COV, the driver adds this property while the point is in COV mode. It contains the value from the source point to be pushed to the cloud via COV update message.
Last Sent To Cloud	read-only timestamp	Shows the date and time the last point update was sent to the

Cloud Alarm Ext (nCloudDriver-CloudAlarmExt)

A Device Extension used for handling alarm routing, specifically acknowledgments for Niagara alarms received from cloud connected applications.

Figure 47 CloudAlarmExt properties



To access these properties, expand **Config→Drivers→NiagaraCloudNetwork→CloudSentienceDevice**, double-click **Alarms**.

Name	Value	Description
Alarm Class	Default Alarm Class (default)	Specifies the alarm routing option for the component.
Last Received Time	timestamp, null (default)	Displays when the system generated the last alarm assigned to this **`Alarm Class`**

Cloud History Device Ext (nCloudDriver-CloudHistoryDeviceExt)

Similar to points, histories can be pushed to the cloud.

Using the Cloud History Device Extension, click **Discover** to show all histories and allow selections to be added the device database for export to the cloud platform. When this is done, you can set a time period

for how often to send that data. All data is queued on the station/Supervisor until that period expires. When it expires, the histories are bundled into 500-record bundles and sent to the cloud platform. The default bundle size is 500, but it is configurable via the History Batch Size property.

Learn Histories action

The automatic **Learn Histories** action provided by the CloudHistoryDeviceExt creates a CloudHistoryFolder for each HistoryDevice it finds in the station's history database, including the local station. This helps organize the potentially large number of history exports into a logical structure. It will also provide some performance improvement for extremely large stations by decreasing the number of children in a single container. The **Learn Histories** action is strongly recommended over standard history discovery and adding, as it will help avoid creating duplicate history exports for a single history.

CAUTION: If you create two cloud history exports for a single Niagara history, you will end up causing overlapping records in the Honeywell Forge database, which will result in missing or incomplete data for cloud-based applications. For this reason, we suggest reducing the likelihood of duplicate exports by using the **Learn Histories** workflow.

Cloud histories and restored connectivity

When the station is disconnected from the cloud for any reason, history records are not pushed to the cloud. After reconnecting, sending history records to the cloud resumes. In order to avoid problems that can arise with excessive bandwidth usage, the driver will NOT necessarily send every record that has not yet been sent. The maxBackfillDuration property describes how far back the driver will look within each history to begin sending records.

For example, consider a history collecting records every 15 minutes at :00, :15, :30, and :45 past the hour, and a maxBackfillDuration setting of 1 hour.

- The station loses connectivity at 11:58pm. Records continue being collected locally at 12:00, 12:15, etc.
- At 3:50, connectivity is restored. There are now 16 unsent records, with timestamps 12:00, 12:15, 12:30, ... 3:30, and 3:45.
- The records that will be sent to the cloud are timestamps 3:00, 3:15, 3:30, and 3:45. The records 12:00 through 2:45 are not sent

The station output will contain one or two log messages whenever cloud history export backfill is limited by this mechanism. The first message indicates that the lastSentToCloud timestamp is being checked on all history exports, using the calculated backfill starting point. For example, after the cloud connector reconnects, you will see a message like this in station output, provided you have FINE or higher configured for the ncloud.history logger:

```
FINE [3:50:56 PM 24-Jan-19 EST][ncloud.history] Checking lastSentToCloud on history exports using backfill start 24-Jan-19 2:50 PM EST
```

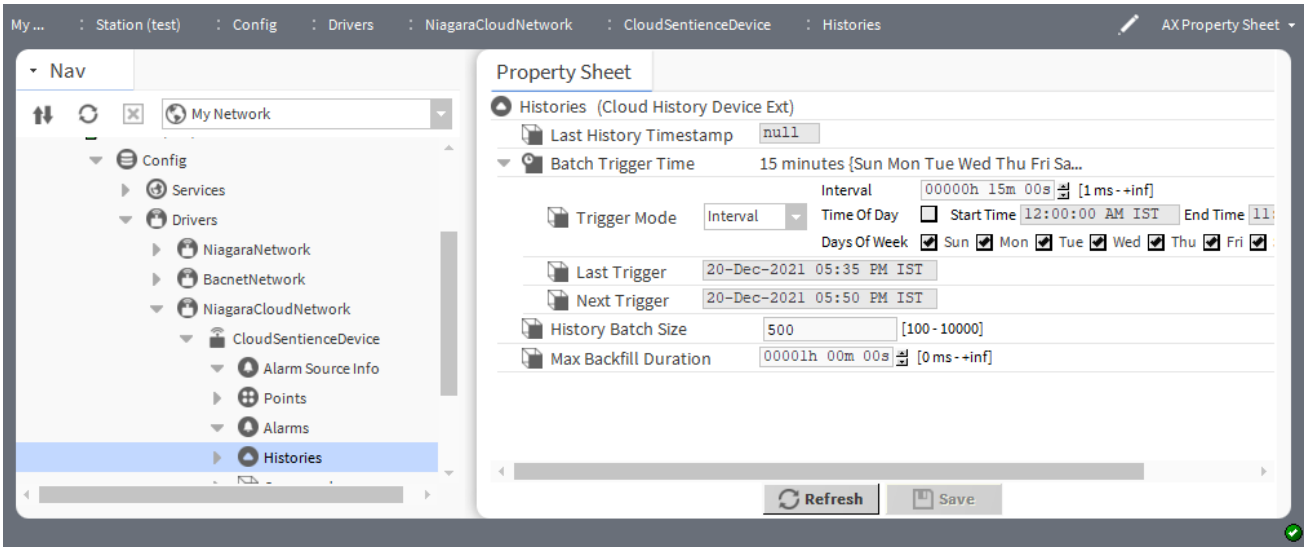
This does not mean anything is being changed, only that the comparison is being made. If no exports exist, or if the lastSentToCloud on all exports is already later than the calculated backfill start time, then nothing further will be logged. If any cloud history exports have a lastSentToCloud that is earlier than the backfill start time, their lastSentToCloud will be updated with this new time. The number of exports modified will be totaled and included in a second log message. For example, you would see a second message like this:

```
INFO [3:50:57 PM 24-Jan-19 EST][ncloud.history] The backfill start time of 24-Jan-19 2:50 PM EST was used to set lastSentToCloud on 17 history exports; history records earlier than this will not be sent to the cloud!
```

Note that the first message will be printed any time the connector connects, including on renewal of the access token. For normal system configuration, this will not result in skipping the export of any history records to the cloud, as the backfill duration is generally much longer than the history export batch trigger interval.

Note that any time the lastSentToCloud is adjusted by the backfill limitation mechanism, the history export's lastSentToCloud property will be set; this may not reflect that a history record was actually sent. For example, if the history export is disabled, its lastSentToCloud may still be adjusted to prevent sending too many records if it is subsequently enabled.

Figure 48 Cloud History Device Ext properties



To access these properties, expand **Config→Drivers→NiagaraCloudNetwork→CloudSentienceDevice**, right-click **Histories** and click **Views→AX Property Sheet**.

Name	Value	Description
Last History Timestamp	read-only timestamp	Shows the date and time the last history record occurred.
Batch Trigger Time	time trigger additional properties (**`Interval`** defaults to 15 minutes, **`Time of Day`**, **`Days of Week`**)	Sets either the history **`Interval`** or individual history items to Batch Trigger mode, but does not set both. For efficiency, use this property with **`Interval`** while setting the individual history items to Manual mode. Leave this property set to its default or greater. Reducing this value causes a warning in the station output. Unintended consequences could occur under some conditions. Obtain guidance from your Support channel prior to adjusting this value. **CAUTION:** Setting the **`Interval`** time to less than 15-minutes may cause several unintended results, such as message throttling by the Sentience IoT Hub and/or prevention of more important messages (such as alarms) from being sent, etc.
Trigger Mode	interval (default), daily, manual	Determines when a TimeTrigger will fire its trigger. Interval fires a trigger each time the specified interval has elapsed. Intended for use when firing the trigger several times per day (e.g. every 5 minutes). Daily allows the trigger to be fired at a specific time on selected days of the week. Also, provides randomization interval so that the trigger is not fired at exactly the same time every day. Manual requires human interaction to fire a trigger.

Name	Value	Description
Batch History Size	500 (default)	Number of history records to include in a single message to the cloud platform. Acceptable range is 100–10,000 records.
Max Backfill Duration	RelTime (default 1 hour)	Specifies the maximum duration of record backfill that will occur when the connector is reconnected after a disconnection. This can be disabled by setting it to a very high number. For example: setting it to 90000 hours (10 years) causes all Cloud Export components to retain the value of "Last Sent to Cloud" which they had before the device lost connectivity. However, this is not recommended in conditions of high-volume batch history exports. **CAUTION:** The Max Backfill Duration should always be more than twice the Batch Trigger Time. (For example: If Batch Trigger Time Interval is adjusted to 30 minutes, Max Backfill Duration should be remain at 1 hour, or more.).

Actions

Following are the three actions:

- **Retry** — downloads the history again.
- **Ping Last History** — Displays the last history record.
- **Execute** — executes the selected history.

Cloud History Folder (nCloudDriver-CloudHistoryFolder)

This component is provided for organizing history exports, if desired. This component is available in the **`nCloudDriver`** palette under History.

The typical usage is to represent a Niagara HistoryDevice in the History Database. The **Cloud History Export Manager** is the default view for this component. Use this view to discover histories inside the station's history database, and add them to the database of history exports to the cloud.

Cloud History Export (nCloudDriver-CloudHistoryExport)

This component defines the export parameters for a local history, including "last sent to cloud time", current status, and remote history Id. **NiagaraHistoryExports** reside under the History extension of the **NiagaraStation** in the **NiagaraNetwork**.

Cloud history exports that are added through the **Learn Histories** job will have their "last sent to cloud time" set. The time that is used is based on the value entered in the learn job configuration window. When the configuration window comes up, by default the value for this time is populated with "now". If the user does not change this value the exports will be created with a "last sent to cloud time" of "now". This is because there is no way of knowing anything about the station doing the export, so "now" is the earliest safe time that can be used. History exports added through **Discover** will not have a "last sent to cloud time" set. There is no constraint on the time (other than that it is stored internally as milliseconds since epoch). For example, you can set your history export to start exporting records at a future date/time, such as starting next week.

For more details, see nCloudDriver-CloudHistoryExportManager, page 114.

Figure 49 CloudHistoryExport properties

To access these properties, expand **Config→Drivers→NiagaraCloudNetwork→CloudSentienceDevice→-Histories**, double-click **Local_LogHistory** (or any of the discovered history type).

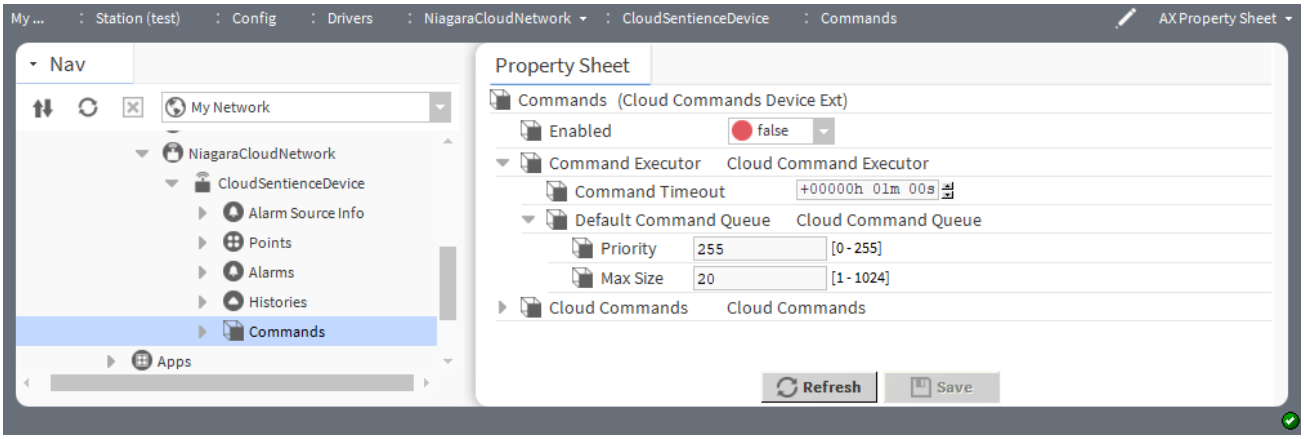
In addition to the standard properties (Status, Enabled, Fault Cause and State), these properties support this component.

Property	Value	Description
Last Attempt	Timestamp	Reports the date and time of the last attempted execution.
Last Success	Timestamp	Reports the last time the station successfully performed this function.
Last Failure	Timestamp	Reports the last time the system failed to perform this function. Refer to **`Fault Cause`** for details.
History Id	text in two parts: `/stationname/historyname`	Specifies the history name in the local station's **History** space, using two parts: "`/<stationName>`" and "`/<historyName>`". If Discovered, station name is "`^`" (a character representing the device name of the parent container) and history name reflects the source history name. Typically, you leave both properties at default values, or edit the second (`<historyName>`) property only.
Point Id	text	Specifies an ID for the point.
Source Ord	File chooser	Chooses the files for the ORD.
Last Sent To Cloud	timestamp	Reports the last history record sent to the cloud. This value is always present. It determines where in the local Niagara history to start from when sending history records to the cloud during batch or individual history data export.

Cloud Commands Device Ext (nCloudDriver-CloudCommandsDeviceExt)

The cloud platform can send system commands down to the JACE/Supervisor. The **CloudCommandsDeviceExt** contains registered command(s) for handling such incoming messages. The appropriate registered command is called to process the message.

Figure 50 Cloud Commands Device Ext properties



To access these properties, expand **Config→Drivers→NiagaraCloudNetwork→CloudSentienceDevice**, double-click **Commands**.

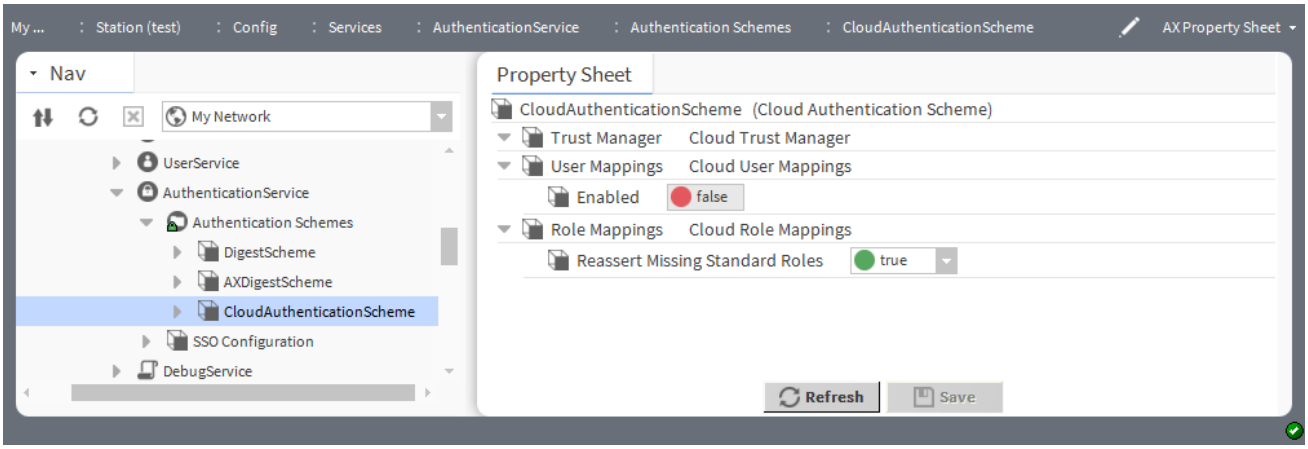
In addition to the standard property Enabled these properties support this component.

Property	Value	Description
Enabled	`true` or `false` (default)	Set to `false` by default, this property activates/deactivates use of system commands.
Command Executor	additional properties	A frozen slot on the **CloudCommandsDeviceExt**, this contains the Cloud Command Executor.
Command Timeout	`00000h 01m 00s` (default)	Specifies how long the executor is willing to let a command run before responding to the cloud with a failure and attempting to cancel the command execution.
Default Command Queue	additional properties	A frozen slot on the Cloud Command Executor, this holds a Cloud Command Queue. Additional Cloud Command Queues can be dragged onto the executor from the **nCloudDriver** palette. Each Cloud Command Queue contains a Priority and Max Size sub-properties.
Priority	`255` (default)	Specifies a priority for the queue. The range is from 0 to 255, where like alarms 0 is highest priority.
Max Size	`20` (default)	Specifies the maximum size limit for the queue. The range is from 1 to 1024.
Cloud Commands	additional properties	Container for custom cloud commands that may be installed on the system. Generally used to view and modify any configuration, such as enable/disable, parameters, etc. For more details, see Custom commands, page 117

Cloud Authentication Scheme (nCloudDriver-CloudAuthenticationScheme)

An authentication scheme verifies that a user is authorized to access a station. Schemes are added to or removed from the **AuthenticationSchemes** container in the **AuthenticationService**. All authentication requests are routed through the system's **AuthenticationService**. This component is available in the **nCloudDriver** palette under **Authentication**.

Figure 51 Cloud Authentication Scheme properties



To access these properties, expand **Config→Drivers→Services→Authentication Service→Authentication Schemes**, double-click **CloudAuthenticationScheme**.

Property	Value	Description
Trust Manager		CloudTrustManager is a container for the added CertTrust-Mapping and JwksTrustMapping components.
User Mapping	`true` or `false` (default)	Available as a read only property, it is not directly editable. Disabled by default. If set to `true`(via Actions), UserMappings are enabled. If `false`, cloud login attempts will fail, throwing the FailedLoginException to inform the user.
Role Mapping	Name you give the component when you add it to the role mappings. For example, "Role-Mapping-Operator".	Matches a cloud role to an actual role on the station.

Cloud Trust Manager (nCloudDriver-CloudTrustManager)

This component is a container for the added **CertTrustMapping** and **JwksTrustMapping** components.

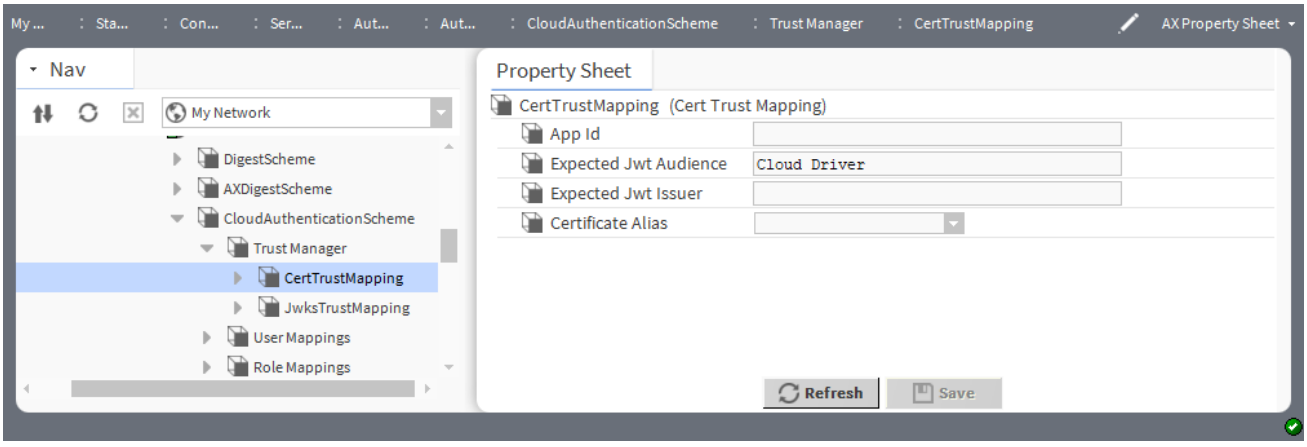
Expand **Config→Services→AuthenticationService→AuthenticationSchemes→CloudAuthentication-Scheme→Trust Manager**.

This container includes no properties of its own to configure.

Cert Trust Mapping (nCloudDriver-CertTrustMapping)

This component is required to configure the station to receive commands sent from the cloud platform. This component is available in the **nCloudDriver** palette under **Authentication**. This component is added to the **Trust Manager** in the **CloudAuthenticationScheme** in the **AuthenticationService**.

Figure 52 Cert Trust Mapping properties



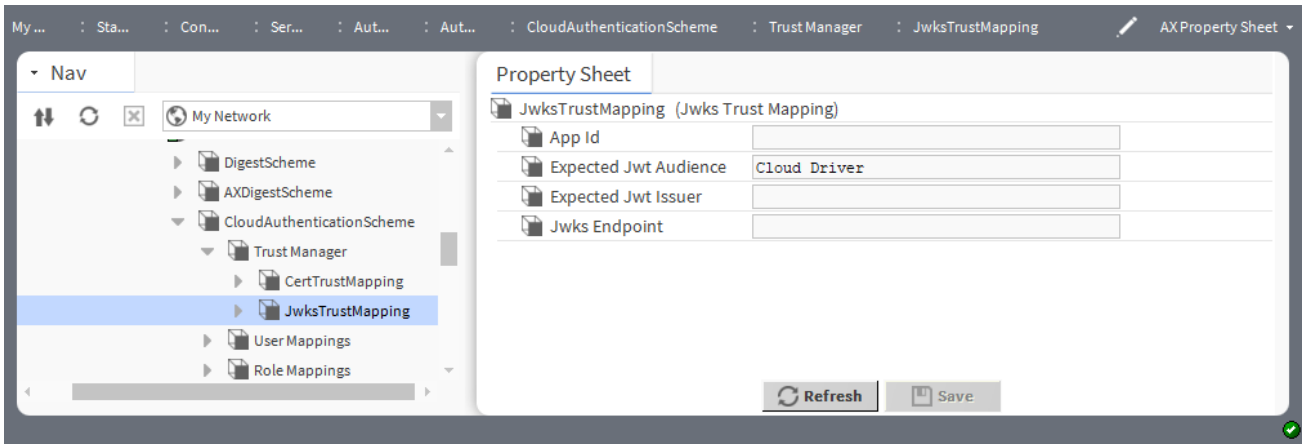
To access these properties, expand **Config→Drivers→Services→Authentication Service→Authentication Schemes→CloudAuthenticationScheme→TrustManager**, double-click **CertTrustMapping**.

property	Value	Description
App Id	text	Value of the Honeywell Forge application id.
Expected Jwt Audience	Cloud Driver (default)	Value of the token audience “aud” field. By default, “Cloud Driver”, but this may be changed to match the value present in the JWT for those providers that do not have a fully configurable audience field. For example, Salesforce prepends the Salesforce application Id (not to be confused with the Honeywell Forge application Id) onto the audience.
Expected Jwt Issuer	text	The URL of the user identity provider. **NOTE:** This value is required. The value for the Token issuer “iss” field must to be provided by the Developer/Integrator during Certificate Trust Mapping configuration.
Certificate Alias	drop-down list	Alias of your token provider's public certificate file that was imported.

Jwks Trust Mapping (nCloudDriver-JwksTrustMapping)

Jwks trust mapping is required to configure the station to receive commands sent from the cloud platform. This component is available in the **`nCloudDriver`** palette under **Authentication**. This component is added to the **Trust Manager** in the **CloudAuthenticationScheme** in the **AuthenticationService**.

Figure 53 Jwks Trust Mapping properties



To access these properties, expand **Config→Drivers→Services→Authentication Service→Authentication Schemes→CloudAuthenticationScheme→TrustManager**, double-click **JwksTrustMapping**.

Property	Value	Description
App Id	test	Defines the Honeywell Forge application ID.
Expected Jwt Audience	text (defaults to Cloud Driver)	Defines the token audience “aud” property. You may change the default to match the value present in the JWT for those providers that do not have a fully configurable audience property. For example, Salesforce prepends the Salesforce application Id (not to be confused with the Honeywell Forge application Id) onto the audience.

Property	Value	Description
Expected Jwt Issuer	text	Typically, defines the URL of the user identity provider. **NOTE:** This value is required. The value for the Token issuer "iss" property must to be provided by the Developer/Integrator during Certificate Trust Mapping configuration.
Certificate Alias	text	Defines the alias of your token provider's public certificate file that was imported.

Cloud User Mappings (nCloudDriver-CloudUserMappings)

In general, "user mapping" is used for Single Sign-On (SSO) to back-end systems, such as a cloud platform. User mapping maps a portal user ID to the user ID of the back-end system.

In Niagara Cloud user mapping is used when an application id from Honeywell Forge is mapped directly to a user (no authentication checks). Disabled by default, this component provides actions to enable/disable user mappings.

WARNING: UserMappings map only a web application to a user. This does not identify what individual user is making the request, so there is no end-to-end authentication or traceability for these requests. **Due to the inherent security risk of not using end-to-end user identification when executing cloud-originated commands on the station, the use of UserMappings is not recommended.** If you enable this feature, a confirmation window appears prompting you to acknowledge that you wish to proceed with the non-recommended configuration.

Figure 54 Confirmation window when enabling user mappings

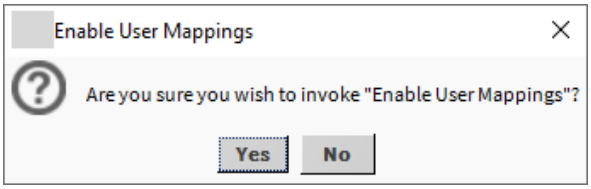
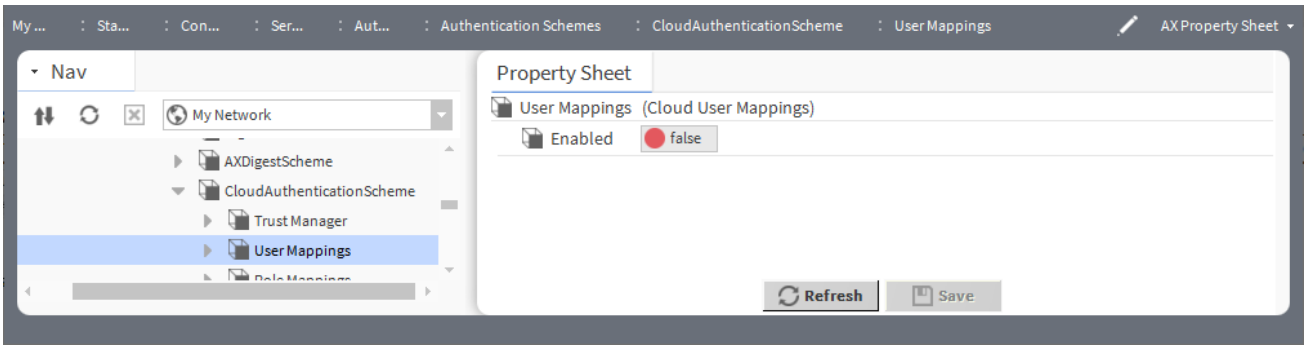


Figure 55 CloudUserMappings property



To access these properties, expand **Config→Drivers→Services→Authentication Service→Authentication Schemes→CloudAuthenticationScheme**, double-click **User Mappings**.

Name	Value	Description
Enabled	true, false (default)	Available as a read only property, it is not directly editable. Disabled by default. If set to "true" (via Actions), UserMappings are enabled. If "false", cloud login attempts will fail, throwing the FailedLoginException to inform the user.

Actions

The following actions are available via right-clicking the component.

- Enable User Mappings
- Disable User Mappings

User Mapping (nCloudDriver-UserMapping)

This component represents an individual user mapping, which is added to the **UserMappings** component in the **CloudAuthenticationScheme**.

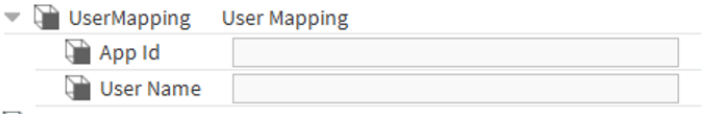
This component is available in the Authentication folder in the nCloudDriver palette.

User Mapping is the mapping of a request's application id directly to a station User. The request is then executed with the permissions context of that station User. User Mapping maps a client application known to Honeywell Forge to a specific station user. It used in the insecure web app id mapping approach to make a connection between the application making the request to Honeywell Forge and a traceable entity known to the station. Incoming requests containing this application id will be executed in the context of the configured station user.

NOTE: User Mapping is strongly discouraged for numerous security and traceability concerns.

If you must use this approach, please note that you cannot map to the station's Admin user. This is explicitly prevented by the driver.

Figure 56 User Mapping properties



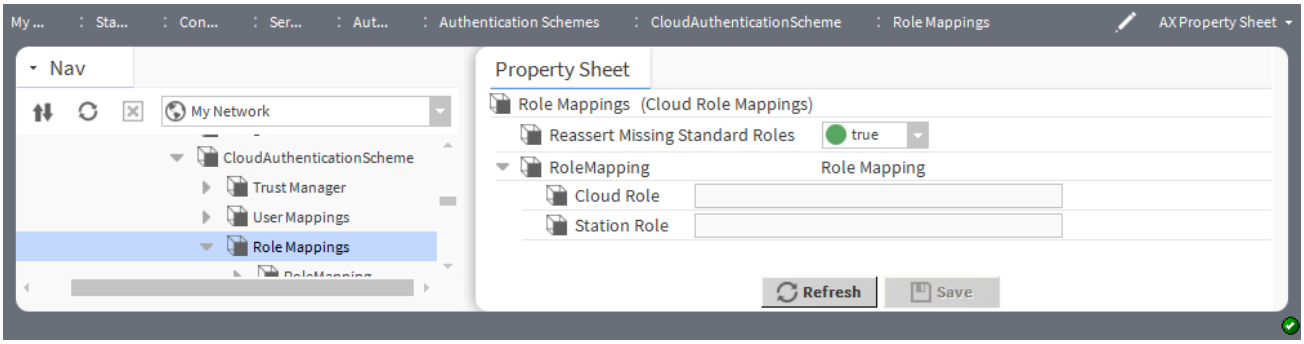
WARNING: UserMappings map only a web application to a user. This does not identify what individual user is making the request, so there is no end-to-end authentication or traceability for these requests. **Due to the inherent security risk of not using end-to-end user identification when executing cloud-originated commands on the station, the use of UserMappings is not recommended.** If you enable this feature, a confirmation window appears prompting you to acknowledge that you wish to proceed with the non-recommended configuration.

Name	Value	Description
App Id	string	Defines the application ID of an application to be mapped.
User Name	string	Defines the name of a user in the **UserService** to associate with requests with this application ID.

Role Mappings (nCloudDriver-RoleMappings)

Roles are used to specify the authorization to station resources for System Commands.

Figure 57 Example Cloud Role Mappings properties



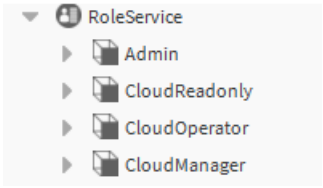
The roles that are authorized in the cloud application are contained in the security token sent with a System Command. These are in a claim called "cloudroles", which is a comma separated list of text strings. For example: "cloudroles": "CloudRole-Manager, CloudRole-Operator". The Role Mappings component provides a way to match the cloud roles to actual roles on the station. So if the cloud role is "CloudRole-Operator", it can be mapped to the role of "CloudOperator" on the station.

Once configured, the station is ready to receive commands with the specified cloud roles. You need to add one role mapping for each cloud role contained in your security token. More than one cloud role can be mapped to the same station role if necessary.

Standard pre-configured roles

The Role Mappings component creates three standard station roles as a convenience. These are CloudReadonly, CloudOperator and CloudManager.

Figure 58 Standard pre-configured cloud roles



By default, each cloud role is “enabled” and has Viewable hierarchies set to “none”. The default values for the permissions of these roles are shown in the following table.

Standard role name	Default permissions
CloudReadonly	`1=r; 2=r`
CloudOperator	`1=rwi; 2=r`
CloudManager	`1=rwiRWI; 2=rwi`

These can be removed if necessary. Any role can be created for use with the Role Mappings component.

To prevent these standard roles from being created upon station start up, set the property **`Reassert Missing Standard Roles`** in the Role Mappings component to “false”.

To recreate these standard roles, set the property **`Reassert Missing Standard Roles`** in the Role Mappings component to “true”. The standard roles will be created upon station start. If the standard roles are already present in the Role Service, they will not be replaced.

NOTE: If the permissions of the standard roles are modified, the modified permissions will remain in effect. If the Standard Role is being used by any RoleMapping in the Cloud Authentication Scheme, then the role will be disabled in that Role Mapping upon the next station start. A warning (like the example shown) is logged in the Application Director when this event occurs:

```
WARNING [12:18:24 28-Nov-18 EST][ncloud.security] Permissions for role
CloudOperator (1=rwi;2=rwiRWI) have been changed from default value of
1=rwi;2=r. The RoleMapping (RoleMapping-Operator) that maps to this role
has been disabled.
```

Name	Value	Description
Reassert Missing Standard Roles	true (default), false	Enables/disables creation of the standard pre-configured roles. Setting this to "false" will prevent the standard roles from being created upon station start. When set to "true" a check is made (only on station startup), and for each of the three standard roles: • if the role is missing, it is created with the default permissions • if the role exists with the default permissions, nothing is done • if the role exists with permissions different than the default, any Cloud RoleMapping that references this role will have the Station Role property cleared. The role itself is not changed. If the role permissions are changed during station operation, nothing is done until the next station start.
RoleMapping	Name you give the component when you add it to the role mappings. For example, "Role-Mapping-Operator".	Matches a cloud role to an actual role on the station.
Cloud Role		Set this property to the exact name of one of the cloud roles specified in the claim in your security token (described in the section, "Standard pre-configured roles").
Station Role	additional properties	Set this property to the exact name of an existing role in the Role service of the station. **NOTE:** Do not enter the default "Admin" role for the Station Role. Any role mapping with a station role of Admin will be ignored for security reasons. For details on standard roles, see the section on "Standard pre-configured roles".

Cloud Device Manager (nCloudDriver-CloudDeviceManager)

This is the default view of the **NiagaraCloudNetwork**. It provides support for discovering and adding cloud devices to the station database, for managing device addresses, and downloading applications to cloud devices.

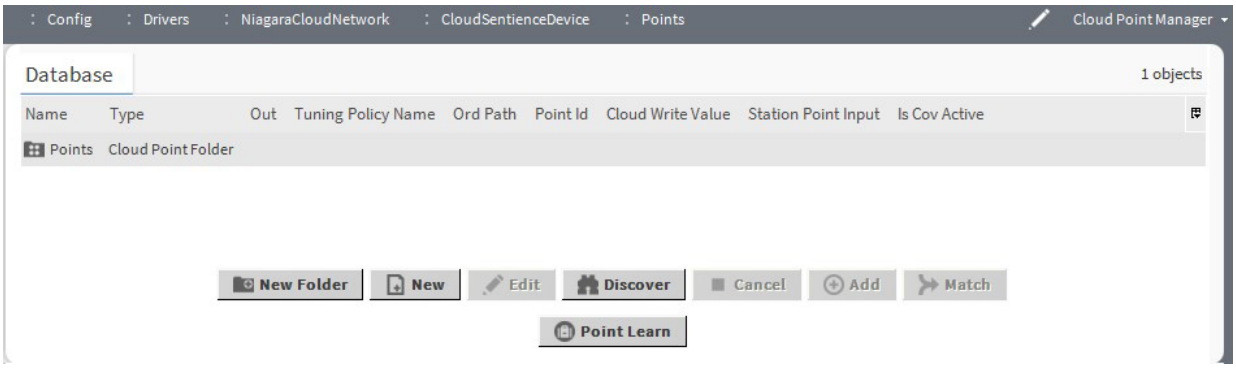
The **Cloud Device Manager** view is built in to the **NiagaraCloudNetwork** component.

Cloud Point Manager (nCloudDriver-CloudPointManager)

The **Cloud Point Manager** is the default view for the Points extension under any CloudSentienceDevice object, and for the CloudPointFolder.

Like other manager views, it is table-based. Each row represents a proxy point (or a points folder) under **Points**.

Figure 59 Cloud Point Manager view



To open this view, expand **Config→Drivers→NiagaraCloudNetwork→CloudSentienceDevice** and double-click the **Points** node.

Columns

Column	Description
Path	Reports the location of the device or event in the station
Name	Reports the name of the entity or logical grouping.
Type	Reports the type of Point.
Out	Reports the out value of the point.
FaultCause	Provides a description of the reason for the fault.
Tuning Policy Name	Reports the name of the selected tuning policy.
Ord Path	Reports the ord path for the selected certificate.
Point Id	Reports the point id.
Is Cov Active	Reports the Cov value.
Last Sent to Cloud	Reports the ltimestamp of the last time a value was sent to cloud.

Buttons

- **New Folder** creates a new folder for devices. Each such folder provides its own set of manager views.
- **New** creates a new device record in the database.
- **Edit** opens the device's database record for updating.
- **Discover** runs a discover job to locate installed devices, which appear in the **Discovered** pane. This view has a standard appearance that is similar to all **Device Manager** views.
- **Cancel** ends the current discovery job.
- **Add** inserts into the database a record for the discovered and selected object.
- **Match** associates a discovered device with a record that is already in the database.

- The **Point Learn** button in this view automatically creates a CloudPointFolder for each folder in the station that it finds. This helps organize the potentially large number of points into some logical structure.

  NOTE: For any station with more than several thousand points, it is strongly recommended that you use **Point Learn** over standard point Discovery and Add. This is because it significantly improves performance for large stations over creating all of the cloud proxy points in the same folder.

  For more details on Point Learn see the procedure, “Configuring a cloud Point Learn job” in this guide.

Cloud History Export Manager (nCloudDriver-CloudHistoryExportManager)

The default view for the Cloud History Folder component, the default view to discover histories inside the station’s history database, and add them to the database of history exports to the cloud.

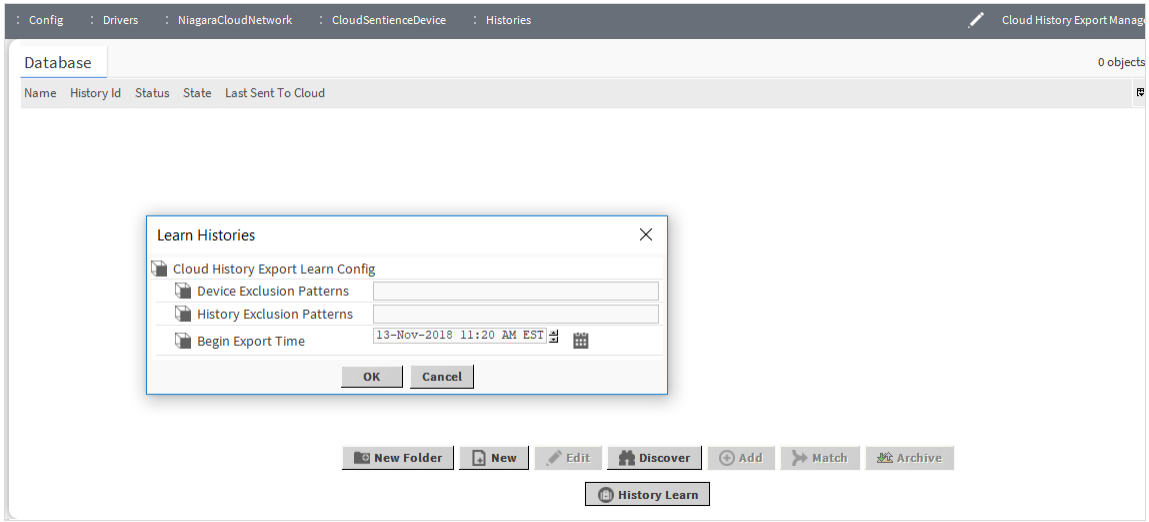
NOTE: In some versions of the driver, you may see the **Archive** button enabled. This button engages the action to individually archive a single history export. However, that action has been disabled, so clicking this button will do nothing. The button will be removed or fully disabled in a future release.

Like other manager views, the **Cloud History Export Manager** is table-based. Each row represents a proxy history (or a history folder) under Histories.

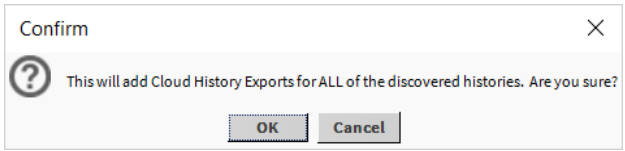
When you use the **History Learn** command, it automatically adds CloudHistoryExports for all the histories in the station's history database. On the subsequent history export attempts it will try to export all records from all of the histories. If you have a station with a very large history database, this could negatively affect communications performance with the cloud. To avoid this situation, an added **Learn Histories** configuration dialog appears which provides the ability to filter the Cloud History Exports that are automatically created from the station's history database.

NOTE: Large history databases could be either a large number of histories (e.g. tens of thousands of histories), or a smaller number of histories with a large number of records. For large history databases, you may want to consider clearing the histories, or clearing older records before adding CloudHistoryExports.

Figure 60 Cloud History Export Manager view with the Learn Histories configuration dialog



When you click OK in the configuration dialog a confirmation dialog appears which allows you to cancel the operation if desired.



Cloud History Export Learn Config properties

The History Learn command provides the ability to filter the Cloud History Exports that are automatically created from the station's history database. This can be helpful if you need to use the History Learn command on a very large station, but want to avoid adding certain histories, or are trying to keep the size down by adding in steps.

Name	Value	Description
Device Exclusion Patterns	string	This is a semicolon-separated list of matcher patterns, which accepts wildcard characters: "?" for one history, and "*" for any number of histories. A HistoryDevice that matches with a pattern in the Device Exclusion Pattern field will not be mapped by a CloudHistoryFolder, and none of the histories below it will be exported.
History Exclusion Patterns	string	A semicolon-separated list of matcher patterns, which accepts wildcard characters: "?" for one history, and "*" for any number of histories. A History with a historyName portion matching one of the patterns in the History Exclusion Patterns field will not be mapped by a Cloud History Export.
Begin Export Time	date/time/time zone	This is an editable timestamp field. The default value is the present time when the "History Learn" button was clicked. This property provides a starting point for the earliest history records to be sent to the cloud. Only history records created after the specified time will be sent to the cloud. You can set your Cloud History Export components to only export history records from the present moment (recommended practice). Or you can set it to export records from a point in time in the past, such as the start of the week or month. Also, you can set it to begin exporting records at a future time only, such as starting at the end of the current month. If the field is left at default, it will begin exports with the current time, it will not send any older data.

Columns

To open this view, expand **Config→Drivers→NiagaraCloudNetwork→CloudSentienceDevice** and double-click the **Histories** node.

Columns

Column	Description
Name	Reports the name of the entity or logical grouping.
History Id	Reports the History Id.
Enabled	Indicates if the network, device, point or component is active or inactive.
Status	Reports the current condition of the entity as of the last refresh: {alarm}, {disabled}, {down}, {fault}, {ok}, {stale}, {unackedAlarm}
State	Reports the current state.
Last Sent to Cloud	Reports the timestamp of the last time a value was sent to cloud.

Buttons

- **New Folder** creates a new folder for devices. Each such folder provides its own set of manager views.
- **New** creates a new device record in the database.

- **Edit** opens the device's database record for updating.
- **Discover** runs a discover job to locate installed devices, which appear in the **Discovered** pane. This view has a standard appearance that is similar to all **Device Manager** views.
- **Add** inserts into the database a record for the discovered and selected object.
- **Match** associates a discovered device with a record that is already in the database.
- **Archive**: Archives the history.
- The automatic **History Learn** command creates a CloudHistoryFolder for each HistoryDevice it finds in the station's history database, including the local station. This helps organize the potentially large number of history exports into a logical structure. It also provides some performance improvement for extremely large stations by decreasing the number of children in a single container. The **History Learn** command is strongly recommended over standard history discovery and adding, as it will help avoid creating duplicate history exports for a single history.

  NOTE: If you create two cloud history exports for a single history, you will cause overlapping records in the Honeywell Forge cloud database, which results in missing or incomplete data for cloud-based applications. For this reason, it is a recommended best practice to use the **History Learn** workflow to reduce the likelihood of duplicate exports.

  CAUTION: When exporting large histories to the cloud it is possible to generate enough data that Honeywell Forge cloud will block the device. To prevent this, the History Learn job can be configured with a **`Begin Export Time`** such that only history records created after that time will be sent to Honeywell Forge.

Event messages and system commands

An event message is a method of pushing data to the cloud. Specific logic triggers an event message. For example, an alarm triggers a NewAlarm event message.

Event messages

The CloudDevice contains a "concrete cloudFactory" to accomplish such things as getting message types to send to the cloud. For example, the SentienceCloudProtocolFactory located within the CloudSentienceDevice creates messages that are specifically formatted to work with the Honeywell Sentience cloud platform. The types of event messages are listed here:

- NewAlarms - sent to the cloud provider
- AlarmAckRequests - from the cloud provider
- AlarmAckResponses - to the cloud provider
- As well as others such as DeviceAckRequests, etc.

System Commands

The cloud platform can send system commands down to the JACE/Supervisor. Handling for these system commands is done by extending a specific class. This is done via the CloudCommandsDeviceExt.

When a system command is received via the CloudConnector, the CloudCommandsDeviceExt is called to determine if it has a registered command to handle the incoming message. If there is one, the appropriate registered command is called to process the message.

Currently, there are 12 built-in commands to read/write to the points that are in the Niagara Cloud. The commands are listed here.

- AlarmAckCommand — Acknowledges an alarm.
- BatchAlarmAckCommand — Acknowledges a list of alarms.
- CloudMultiPointClearCommand — Releases the values and priority levels of a list of cloud accessible points in the station that were previously set with a CloudPointWriteCommand or CloudMultiPointWriteCommand.

- CloudMultiPointReadCommand — Returns the values of a list of cloud accessible points in the station.
- CloudMultiPointReadInputsCommand — Reads the active inputs to a list of cloud proxy points.
- CloudMultiPointWriteCommand — Sets the values of a list of cloud accessible points in the station.
- CloudPointReadCommand — Returns the values of an individual cloud accessible point in the station.
- CloudPointReadInputsCommand — Reads the active inputs to a cloud proxy point.
- CloudPointWriteCommand — Sets the value of an individual cloud accessible point in the station.
- CovActiveCommand — Sets a list of points into COV mode.
- CovInactiveCommand — Takes a list of points out of COV mode.
- InvokeCommand — Allows invocation of an action on a component.
- RetrieveCloudCommandsCommand — Lists the names of all commands that are available on this system, including the built-in ones, and any custom commands that are installed.
- RetrieveCloudPointsCommand — Lists the names of all the points in the station that are accessible from the cloud.
- SubscribePointsCommand — Tells one or more cloud proxy points to put their associated station point into subscription.
- UnsubscribePointsCommand — Tells one or more cloud proxy points to put their associated station point out of subscription.

With Cloud Command Queues the JACE/Supervisor now responds as soon as the command has been placed in the queue. When the command executes its output is sent to the cloud via NewEventMessage(s) and when the command exits another NewEventMessage is sent.

The CloudCommandsDeviceExt must be enabled before any individual command can run. Also, the PointReadAllowed and PointWriteAllowed properties for the NiagaraCloudNetwork must be set to “true” in order to service point-reads and point-writes, respectively.

Custom commands

In addition to providing many “built-in” commands, the Niagara Cloud also provides the capability to handle custom commands. This ability to invoke custom code provides greater flexibility, however, there are multiple restrictions and security requirements.

CAUTION: There are known security risks in executing cloud-to-device commands. Refer to the “Chapter 5 Security recommendations, page 39” chapter in this guide for detailed information on design recommendations and security requirements for implementing Cloud Commands functionality on the station.

Glossary

AMQP	Advanced Message Queuing Protocol (AMQP) is an open standard for messaging middleware. It provides a method for connecting applications, across LANs and WANs. AMQP is a wire-level protocol plus a model for routing and queuing messages. It covers very high performance publish-subscribe patterns and high-reliability messaging. The IANA-assigned port number for AMQP is 5672 (TCP, UDP, SCTP protocols). The assigned port number for AMQPS (AMQP protocol over TLS/SSL) is 5671.
Asset Manager	The Asset Manager tool is a web-based tool in Niagara Community which provides a means of managing users, organizations, and devices in preparation for interacting with any of the Niagara Cloud Suite. Additionally, the Asset Manager provides automated software maintenance management and notifications.
concrete CloudDevice	A concrete CloudDevice (CCD) is a class-based programming term that refers a factory method of creating objects which adds flexability to the system. Simply put, a concrete CloudDevice is a built in Extensiblity point. In the case of the Niagara Cloud, every CloudDevice is formatted to work with a specific cloud platform. Inside the CloudDevice is a certain type of factory used to create messages that are formatted to work only with that cloud platform. While another CloudDevice would contain a different type of factory that creates messages specifically formatted to work with a different cloud platform.
concrete CloudFactory	A concrete CloudFactory is a class-based programming term that refers a factory method of creating objects which adds flexability to the system. Simply put, a concrete CloudFactory is a built in Extensiblity point. For example, within the CloudSentienceDevice is a SentienceCloudProtocolFactory that is used to create the event messages that this CloudDevice can send. These event messages are specifically formatted to work with the Sentience Cloud platform. Another CloudDevice would contain a different type of CloudFactory that creates messages specifically formatted to work with a different cloud platform.
GUID	A globally unique identifier (GUID) is a system ID number assigned to a device during the device registration process in the Asset Manager tool.
IoT Hub	A term referencing the Azure IoT Hub which is a Microsoft managed service that enables reliable and secure bidirectional communications between IoT devices and the back-end solution provided by Niagara Cloud. IoT stands for "Internet of Things", and describes the system of interconnected devices that share information across networks all over the world. For more information, visit the Azure website at this address: *https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-what-is-iot-hub*.
JWKS	JSON Web Key Set (JWKS) is a set of keys containing the public keys used to verify any JSON Web Token (JWT) issued by the authorization server and signed.
JWT	JSON Web Token (JWT) is an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. JWT is a standard, meaning that all JWTs are tokens, but not all tokens are JWTs.
Niagara Community	Niagara Community is a portal for the cloud-based services and tools available at: *http://www.niagara-community.com*.

registered device	This term refers to an embedded JACE controller registered with the Asset Manager tool and assigned a global identifier for purposes of tracking software maintenance management and authenticating with Niagara Cloud Suite.
the cloud	Simply put, the term refers to a type of Internet-based computing that provides shared computer processing resources and data to computers and other devices on demand. It is a model for enabling on-demand access to a shared pool of resources (e.g., computer networks, servers, storage, applications and services). Cloud computing and cloud storage solutions provide users and enterprises with capabilities to store and process their data in third-party data centers. In traditional computing systems, users run software applications installed on a physical computer or corporate server. Cloud computing enables users to access the same kinds of applications through the Internet, paying a subscription fee for individual services, thereby paying only for what they use. If an organization quickly needs access to more resources, it can scale quickly in the cloud. If it needs to scale down resources, it can do so just as easily.

Index

O

P

R

S

T

U

V

W