Technical Document

Niagara AWS Utils Guide

March 3, 2023

niagara4

Niagara AWS Utils Guide

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

Confidentiality

Trademark notice

Copyright and patent notice

Contents

About this guide

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

Product Documentation

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

Document Content

This document describes how to move data to AWS IoT via MQTT drivers, provision MQTT connectivity on a fleet of controllers without individual manual setup, and how to renew onboarding for every controller before expiration.

Document change log

Changes to this document are listed in this topic.

- Initial release publication: March 3, 2023

Related Documentation

- Niagara Signing Service Guide
- Abstract MQTT Driver Guide
- Niagara Station Security Guide

Chapter 1 About AWS Utils

Topics covered in this chapter

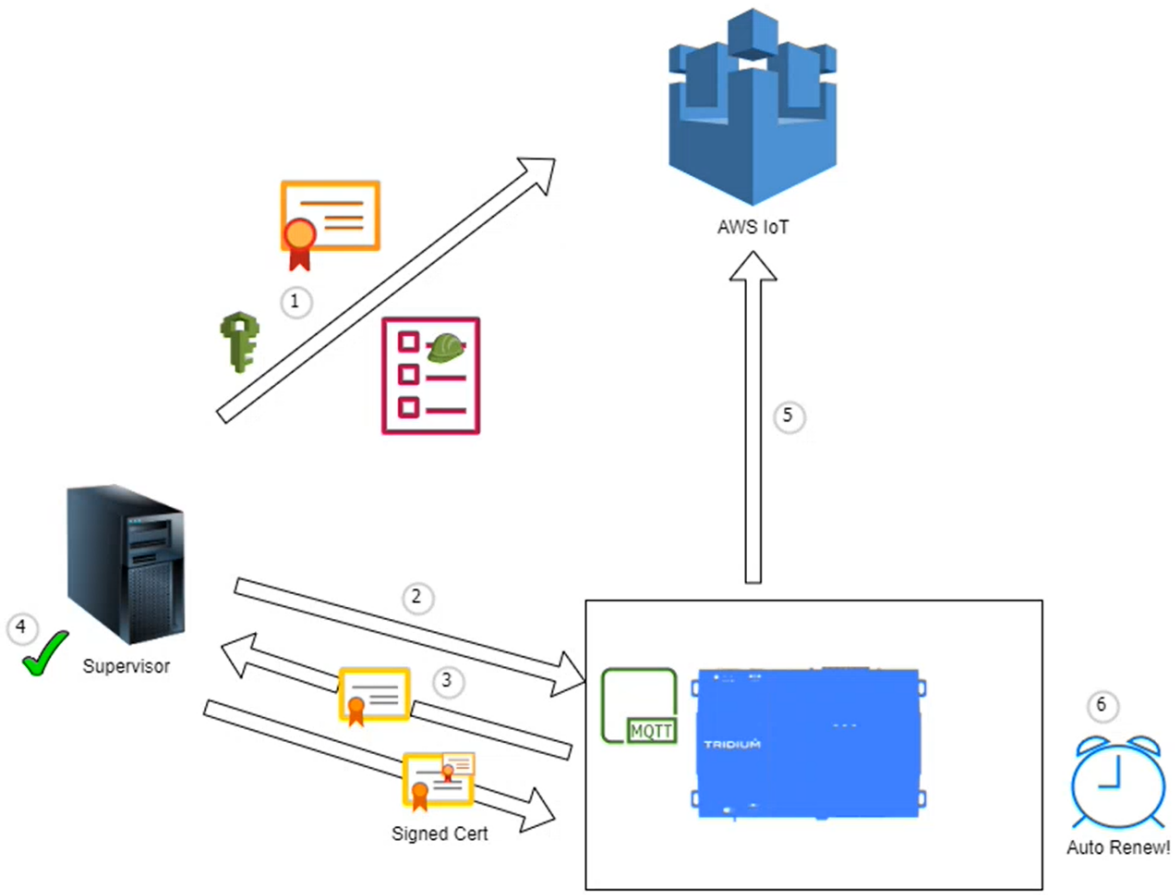
♦ Overview

Overview

As of Niagara 4.13, you can use a simplified and quicker way to publish data to AWS IoT via MQTT drivers, provision MQTT connectivity on a series of controllers with zero touch, and renew onboarding for every controller when they come to expire.

Here are the simplified steps that will save you time:

1. Supervisor: Job creates and registers CA (Certificate Authority).
2. Provision each controller with the new MQTT device.
3. Automatically request a certificate from the Signing Service.
4. Admin user approves request. Certificate is signed (sealed) and delivered.
5. Connect to AWS IoT.
6. Certificate will be auto-renewed before expiry.



Chapter 2 AWS Utils

Topics covered in this chapter

♦ Provisioning NiagaraNetwork with AWS MQTT devices
♦ Configuring Just In Time Provisioning (JITP)
♦ Configuring Signing Service for AWS
♦ Running Install AWS MQTT Device task

The AWS (Amazon Web Services) Utils module provides utilities and configuration to enable a Niagara station to interact with certain services within AWS. New features enable a fleet of Niagara controller stations to be automatically commissioned with MQTT devices and signed device certificates, which automatically connect to AWS requiring no manual user setup on the controller station. Those devices will also automatically renew their certificates prior to expiration and remain connected.

Use:

- Using the service within the module, you can register access keys for authenticating with AWS.
- You can configure and execute service specific tasks, such as interaction with the AWS REST endpoints. These tasks may typically be one-off configuration tasks during station commissioning. Thus, by default access keys are only stored temporarily.

This module also supplies a Niagara provisioning task to furnish the various stations of your NiagaraNetwork with an AWS MQTT device that can utilize the benefits of **Just In Time Provisioning** (JITP).

Prerequisites:

- You have installed the Niagara 4.13 version.
- You have installed the following modules: **`awsUtils-rt`**, **`awsUtils-wb`**, **`awsUtils-ux`**.
- The station hosting the AWS Service must have access to *amazonaws.*.com* on **`port 443`** to interact with the REST endpoint.
- All remote target stations require to be at Niagara 4.13 or higher.

Provisioning NiagaraNetwork with AWS MQTT devices

There are three main steps to create a fleet of MQTT devices on your controller stations:

Prerequisites:

All remote target stations require to be at Niagara 4.13 or higher.

Step 1 Register your account to use the **Just In Time Provisioning** task using the AWS Service on a Supervisor. See Configuring Just In Time Provisioning (JITP), page 9 for more details.

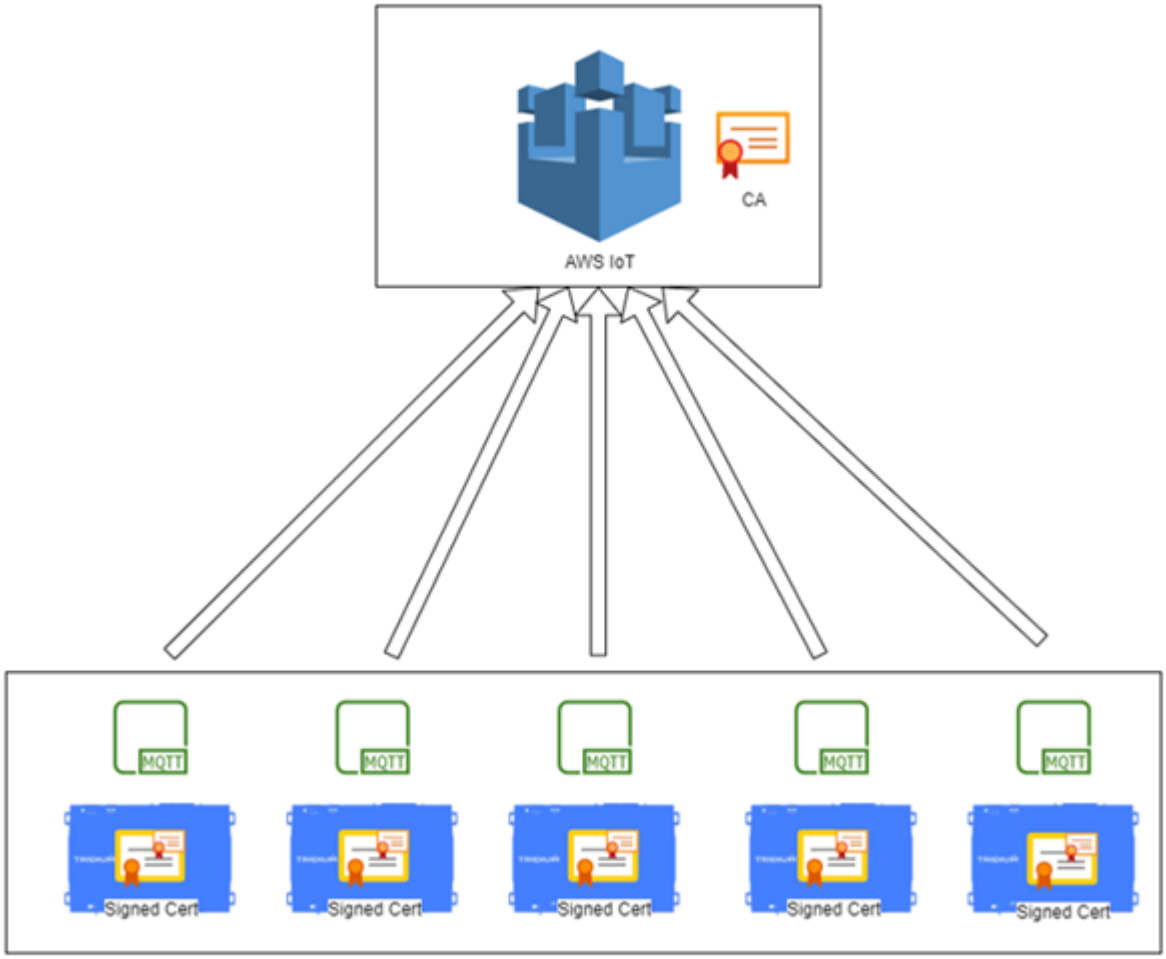
Step 2 Set up the Niagara Signing Service on a Supervisor to supply signed device certificates. See page: Configure the Signing Service for AWS, page 15 for more details.

Step 3 Run the **Install AWS MQTT Device** task to install MQTT devices on each controller. See page: Running the AWS MQTT Provisioning Task, page 16 for more details.

Configuring Just In Time Provisioning (JITP)

AWS Just In Time Provisioning (JITP) allows you to provision your devices within AWS IoT upon first connection, thereby eliminating many manual configuration tasks. This is relevant in that the MQTT driver supports connections to AWS IoT, using certificates as means of authentication for each MQTT device.

The scenario is that you have many controller devices, which require a connection to AWS IoT. Consequently, each controller requires its own certificate signed by a trusted CA certificate*. These certificates in the past have required manual generation and setup within both Niagara and AWS, and a repetition of the same task when they expire.



Commissioning controllers without JITP

The following minimum configuration is necessary to commission a network of controllers **without JITP**.

NOTE:

*In 2022 it became possible to communicate with AWS IoT over MQTT without the requirement for a pre-registered CA certificate, however we recommend to use trusted certificates for security purposes.

Step 1 Set up IAM user.

Step 2 Create an AWS IoT policy for your devices.

Step 3 Attach various allowed actions and roles to the policy.

Step 4 Obtain a CA certificate.*

Step 5 In the AWS console or command line, obtain a **verification code**. *

Step 6 Generate a verification certificate with the verification code as the **Common Name**.*

Step 7 Sign the verification certificate with the CA certificate.*

Step 8 Upload CA certificate and verification certificate.*

Step 9 Activate certificates.*

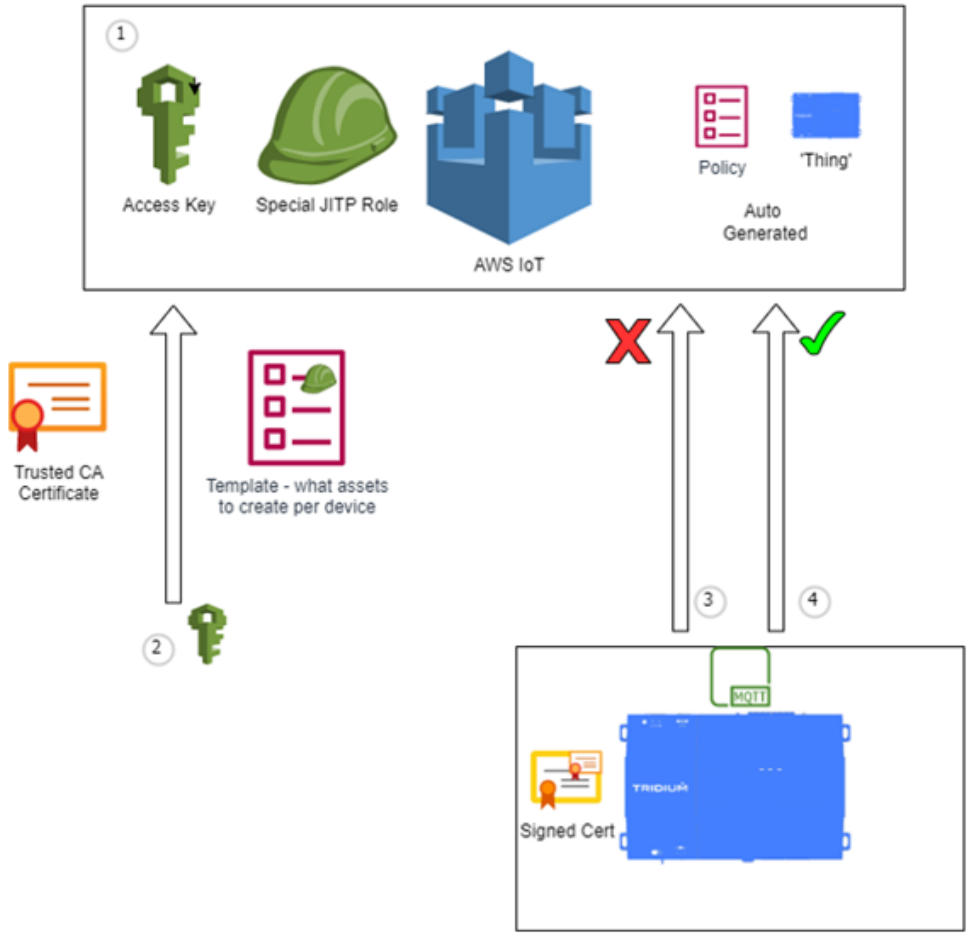
Repeat the following steps for each device:

1. Install the MQTT network and device in your station.
2. Create a **Thing** to represent the device in AWS.
3. Associate IoT policy with the **Thing**.
4. Generate a device-specific client certificate.
5. Sign the device certificate with the CA certificate.
6. Upload the device certificate to AWS and activate.
7. Associate certificate with the **Thing**.
8. Combine CA certificate, device certificate, and key into a `.pem` file.
9. Upload the `.pem` file to the Niagara platform certificate manager.
10. Set the alias of the imported certificate on the MQTT device authenticator.
11. Connect.

Prior to device certificate expiration, it is necessary to repeat these steps for each device .

Commissioning controllers with JITP

The following process describes how you can commission controllers using Just In Time Provisioning.



1. The **`AWSIoTThingsRegistration`** role is assigned to an AWS access key.
2. A CA certificate is registered along with a provisioning template policy by a process authorized by that role.
3. Any device trying to connect to the AWS IoT endpoint for that account will initially have the connection rejected.
4. However, if that certificate presented as authentication was signed by the CA registered in step 1, AWS will then automatically commission a Thing to represent your device according to the rules in the provisioning template, thereby using values from the certificate. When the device makes a subsequent connection attempt, it will be granted access.

The functionality within the AWS service allows a Niagara Workbench user to perform the task of optionally generating a new CA certificate or using a user-imported one, and performing the CA registration and template creation with AWS via their REST API.

The provisioning template created by Niagara Workbench will result in the following:

- The Thing name within AWS matches the **`Common Name`** value of the device certificate.
- The Thing will be granted a policy allowing the following permissions for all resources:
  - iot:Connect
  - iot:Publish
  - iot:Subscribe

- iot:Receive
- iot:GetRetainedMessage
- iot:ListRetainedMessages
- iot:RetainPublish

Setting up JITP for devices

The following steps describe how to set up Just In Time Provisioning for Niagara devices.

Prerequisites:

In AWS IoT:

1. Create an IAM user with an access key and save the key value for future step.
2. Create an IAM service role with the following permissions: `AWSIoTThingsRegistration`, `AWSIoT-Logging`, `AWSIoTConfigAccess`, `AWSIoTRuleActions`.
3. Copy role ARN string for future use.
4. Create an IAM policy and use the following JSON, substituting the value of your ARN string from the previous step:

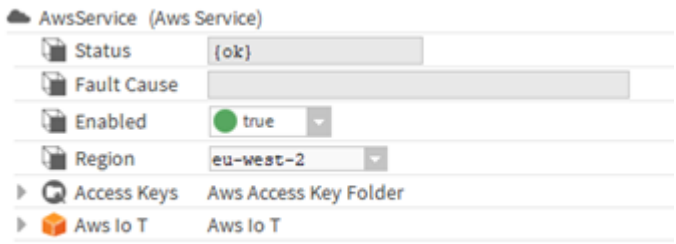
```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iam:GetRole",
            "iam:PassRole"
        ],
        "Resource": "yourArnString"
    }]
}
```

5. Add the policy to your IAM user.

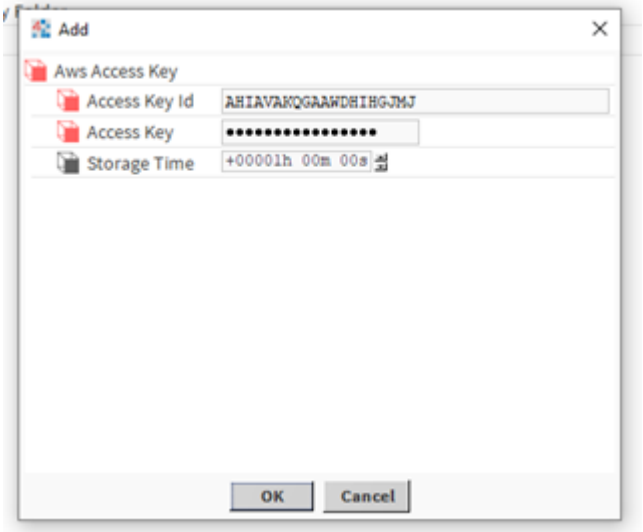
In Workbench:

Step 1 From the **awsUtils** palette, add an **AWS Service** to the **Services** container

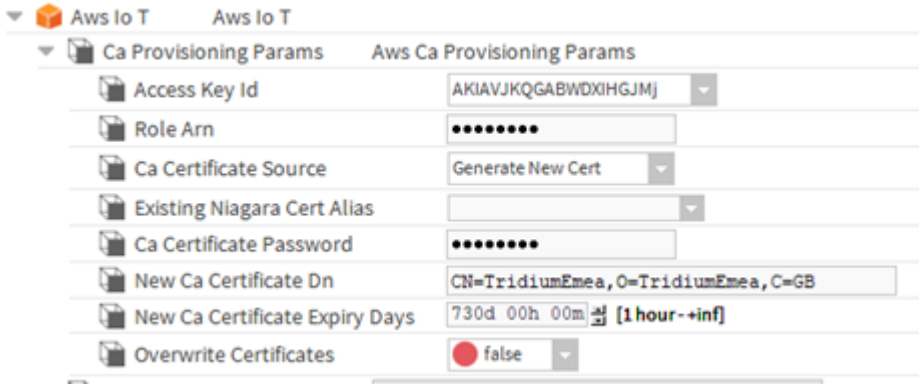
Step 2 Select the AWS region to which your devices will be provisioned.



Step 3 Right-click **Access Keys** and select **Actions→Add** to add the access key, which has the correct permissions configured above.



Step 4 Expand the **Aws IoT** component and populate the **Ca Provisioning Params**.



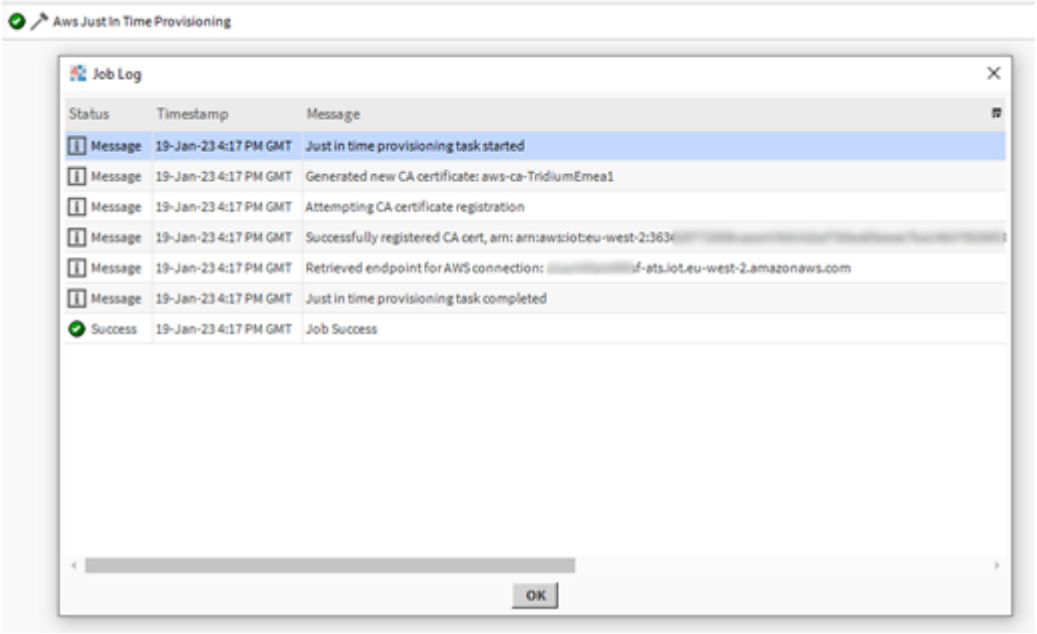
a. Select the access key entered above, and enter the **`Role Arn`** from the previous steps.

b. If selecting the option for Niagara to generate the CA certificate, you will need to populate the **`New Ca Certificate DN`**. If choosing to use an existing CA, import this into the platforms Certificate Management prior to this step. Then pick the alias in **`Existing Niagara Cert Alias`**.

c. Regardless of your CA source, you will also need to enter the password for the CA certificate.

Step 5 Right-click **Aws IoT** and select **Actions→Setup Just In Time Provisioning**.

The Niagara job will be initiated and a **Job Log** window will appear in Workbench.



The **`Mqtt Data Endpoint`** property of your **Aws IoT** component will not be populated with the endpoint that your MQTT devices can use to connect to IoT.

Step 6 You can now install your MQTT devices to communicate with AWS. If you choose to do this manually on each station, you will need to generate each device certificate and sign them individually with the CA. However with Workbench, it is now possible to automate this. See Provisioning NiagaraNetwork with AWS MQTT devices, page 9 for more details.

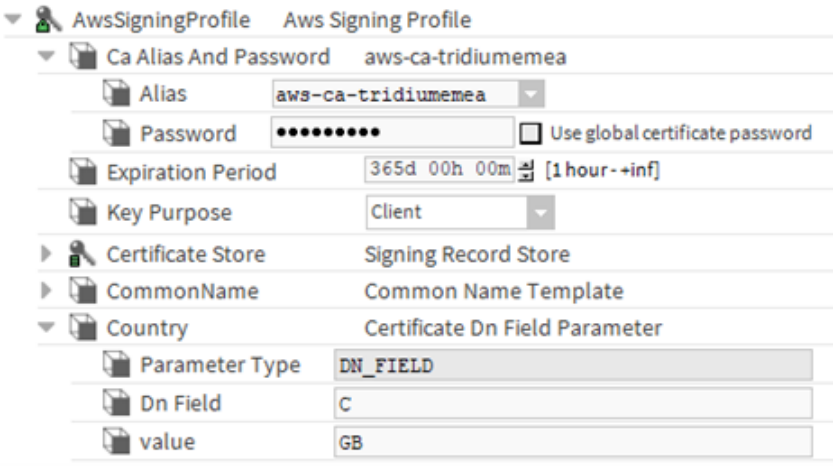
Configuring Signing Service for AWS

This procedure enables the AWS MQTT devices to request signed certificates from the Supervisor.

Step 1 From the **`SigningService`** palette, add a **Signing Service** to your station's **Services** container.

Step 2 From the **`awsUtils`** palette, drag **AwsSigningProfile** to the **Profiles** folder.

Step 3 Expand **Ca Alias And Password**, select the **`Alias`** from the drop-down menu, and enter the **`Password`** for the CA certificate. You may have generated or imported this during JITP setup.



Step 4 To generate a unique certificate **`Common Name`** for each device, you can optionally change the settings in the **`Common Name Template`**.

Step 5 Expand the **`Country`** certificate parameter and enter a value. You can add more certificate parameters from the **`SigningService`** palette, but **`Country`** is a minimum requirement.

To learn more about the Signing Service, see .

Running Install AWS MQTT Device task

This task installs an MQTT device on each remote station, which will automatically onboard itself with the Supervisor's Signing Service, obtain a signed certificate, and connect to AWS.

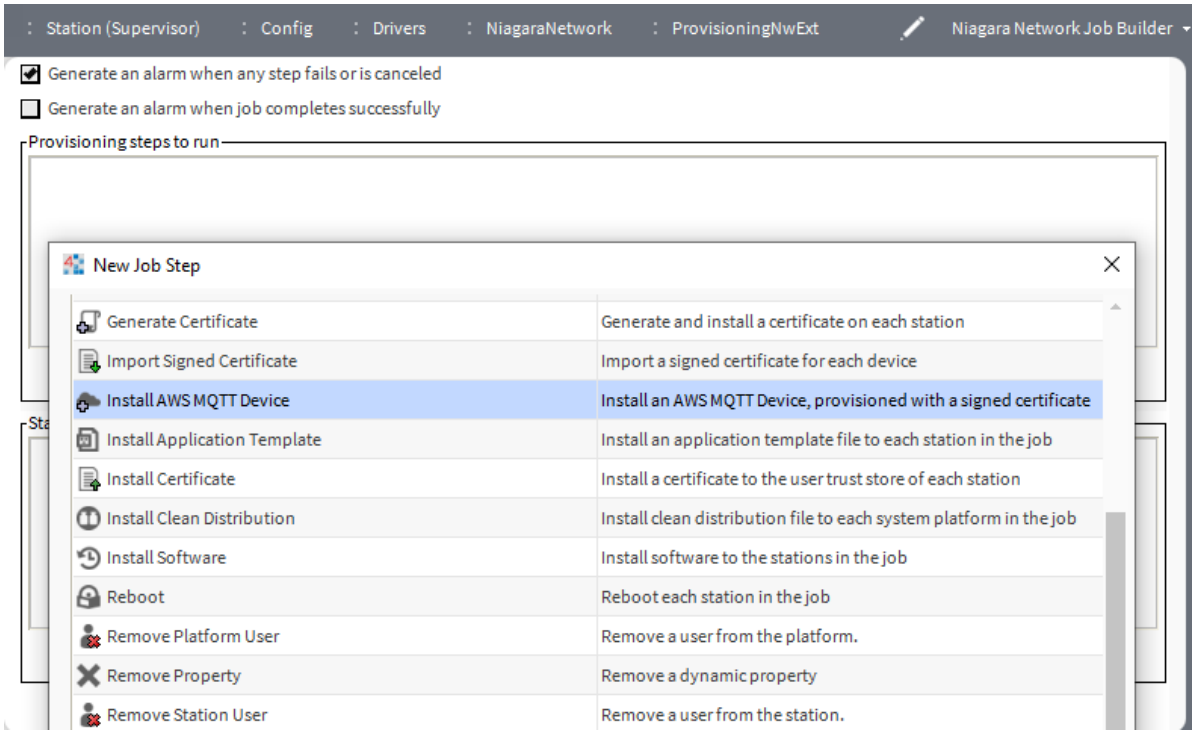
Prerequisites:

- You have an active NiagaraNetwork connection to each station on your Supervisor.
- Each station has a reciprocal connection back to the Supervisor.
- You have completed the previous steps to configure the AwsService and SigningService

Step 1 Navigate to your NiagaraNetwork and double-click **ProvisioningNwExt**.

Step 2 Optionally, you can add the provisioning step **Setup Reciprocal Connection** to create the required connection back to the Supervisor if the stations do not have this already.

Step 3 Add the provisioning step **Install AWS MQTT Device**.



The **Install AWS MQTT Device** windows opens.

Step 4 Fill in the following details in the window:

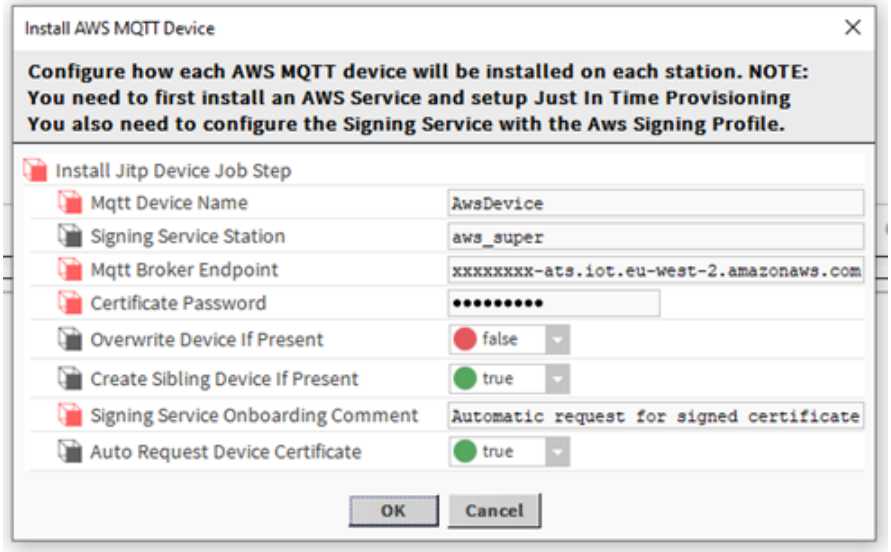
a. Name of the MQTT device. It will be uniform on each station.

b. Name of the Supervisor that contains the Signing Service.

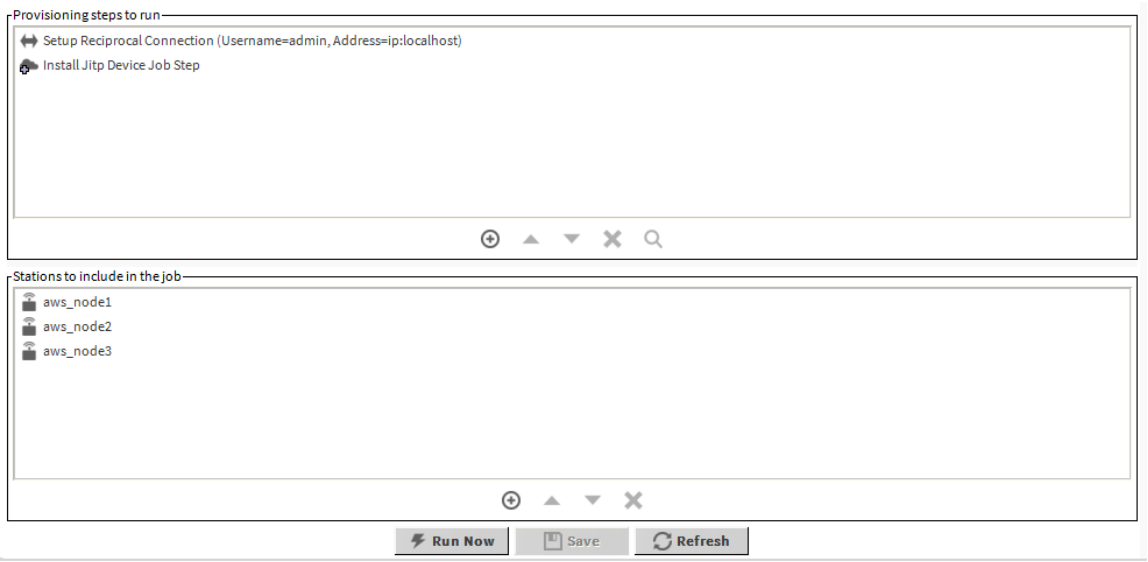
c. Your AWS MQTT endpoint. If you have run the JITP setup, this will be pre-populated.

d. The password to use when storing your signed device certificate.

e. A comment, which will appear in the Signing Service against each device's CSR.



f. Add the stations you wish to provision and click **Run Now**.



Step 5 Navigate to **Signing Service→Transports→foxTransport→Session Token Store** and approve the requests received shortly from each device.

Now each created device on each station automatically obtains their signed certificate and connects to AWS.

Chapter 3 Components, views and windows

Topics covered in this chapter

♦ Components

The user interface includes components, views and windows, which provide the means for communicating with the system.

The Help topics include context sensitive information about each component and view, as well as information about individual windows.

Components

Components include services, folders and other model building blocks associated with a module. You may drag them to a property or wire sheet from a palette.

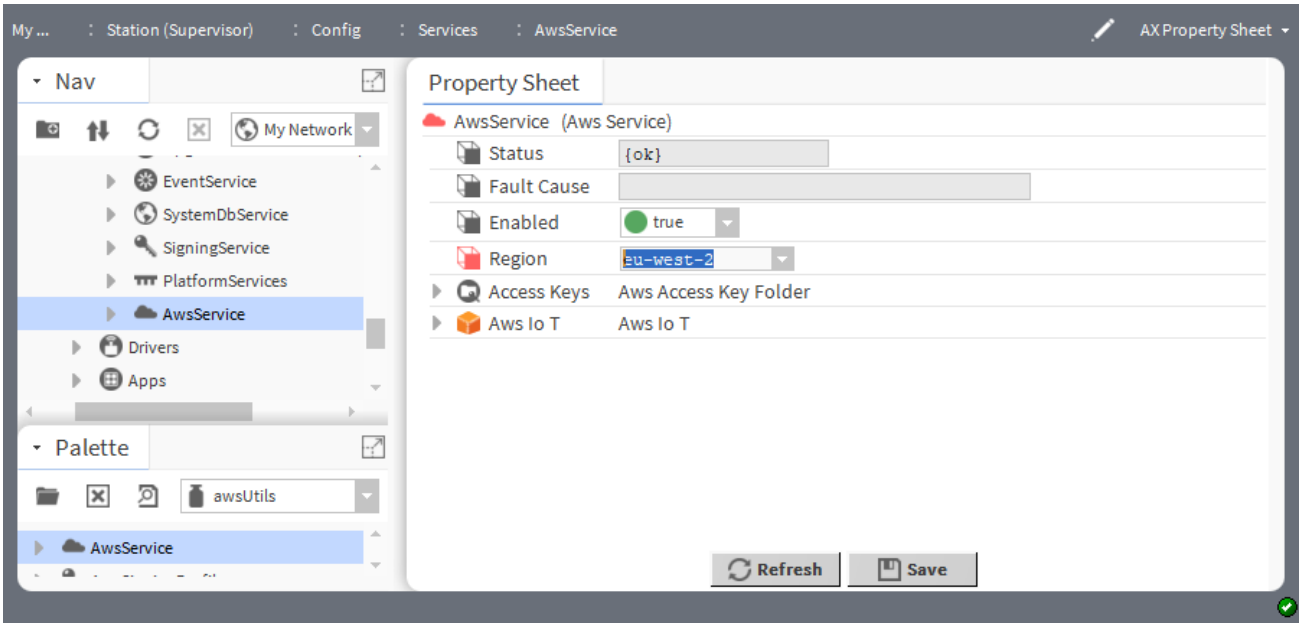
Descriptions included in the following topics appear as context-sensitive help topics when accessed by:

- Right-clicking on the object and selecting **Views→Guide Help**
- Clicking **Help→Guide On Target**

Aws Service (awsUtils-AwsService)

The **Aws Service** provides utilities and configuration to enable a Niagara station to interact with certain services within AWS.

The service itself allows you to define the AWS region to which your assets belong for this station or site. The region is used when forming requests to the REST API. The spy page for the service contains details of REST requests made.



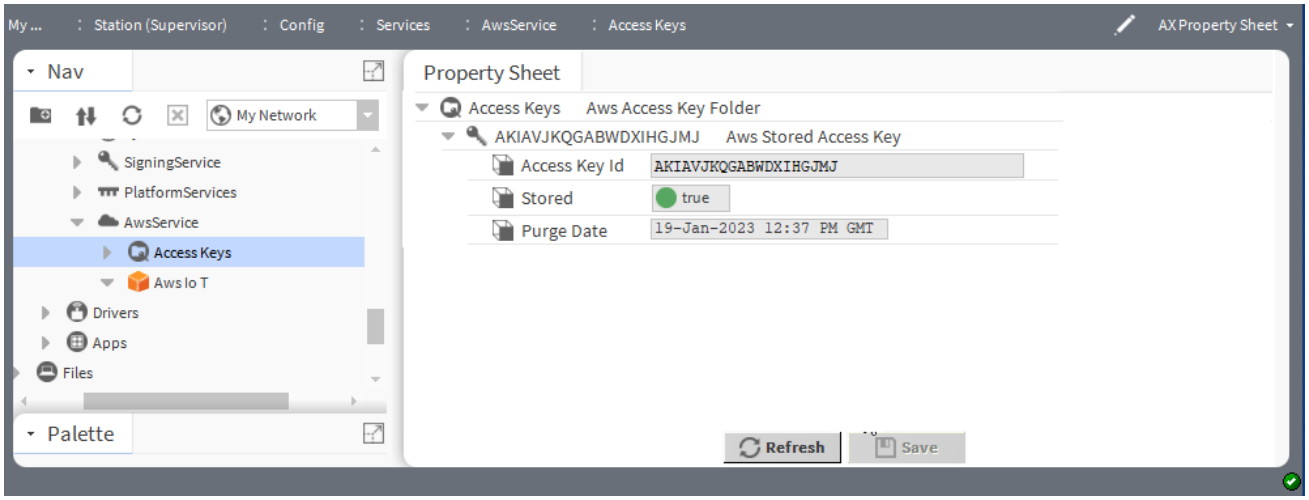
To use the **Aws Service** component, drag an instance from the **awsUtils** palette to the station's **Services** container.

Property	Value	Description
Enabled	`true` or `false` (defaults to `true`)	When false, no external requests will be made by the service or its children.
Region	drop-down menu	Contains the list of AWS regions. Select the one in which your AWS assets and services belong for this station or site.
Access Keys	folder	Holds references to AWS Access Keys.
Aws IoT	additional properties	Contains properties for setup and execution of tasks to configure your AWS IoT service.

Aws Access Key Folder (awsUtils-AwsAccessKeyFolder)

To utilize some of the capabilities of the AWS Service, it is necessary to use an access key for authentication. These tasks may typically be one-off configuration tasks during station commissioning. Thus, by default access keys are only stored temporarily. This is in line with best practice as access keys should only be stored as long as you require them.

AWS Access Keys are stored securely in a dedicated encrypted key ring to which only the **`awsUtils`** module code has access. Keys are automatically purged from the key ring once their storage time has elapsed.



To add an access key, invoke the **Add** action, paste the value in the **`Access Key Id`**, and select the period you wish for the station to securely store the key value. It is best practice to only store the key for the time in which it is required to configure the remote service in AWS.

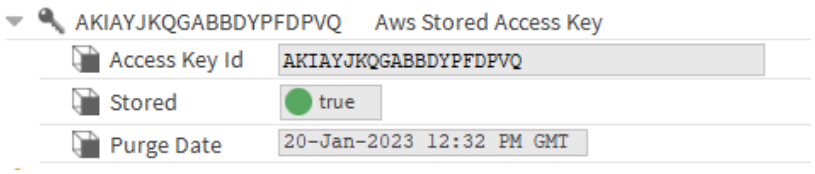
Actions

Add: Adds a new AWS access key

Cleanup Purged Keys: Forces removal of any access keys that have passed their purge time.

Aws Stored Access Key (awsUtils-AwsStoredAccessKey)

Aws Stored Access Key represents an AWS access key securely stored in a dedicated key ring.



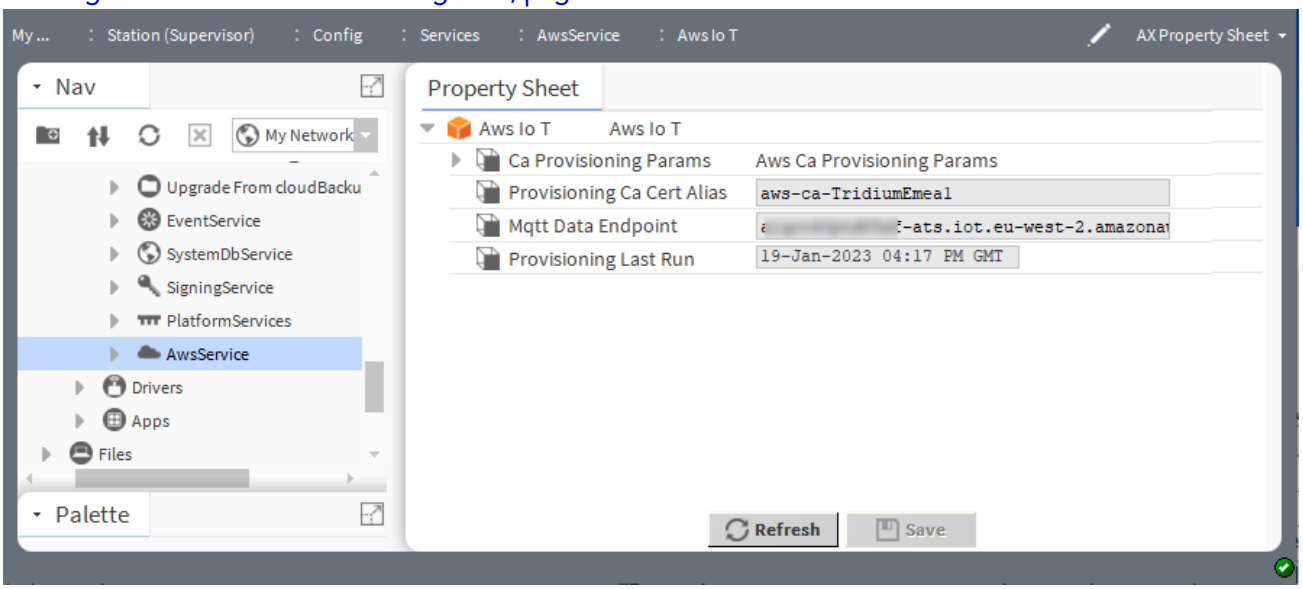
To add an access key, invoke the **Add** action on the **Aws Access Key Folder**, paste the value in the **`Access Key Id`**, and select the period you wish for Workbench to securely store the key value. It is best practice to only store the key for the time in which it is required to configure the remote service in AWS.

Property	Value	Description
Access Key Id	read-only	Displays the unique Id of the access key.
Stored	read-only	If `true`, the key is stored in the key ring. If `false`, the access key is purged.
Purge Date	read-only	Displays the date and time at which the key will be purged from the key ring.

Aws IoT (awsUtils-AwsIoT)

Aws IoT is a component used to set up and execute tasks for the configuration of your AWS IoT service.

After configuring the child **Ca Provisioning Params**, use the **Setup Just in Time Provisioning** action to execute a job to register a CA with AWS. The results of the task are populated in the properties below. See Running the AWS MQTT Provisioning Task, page 16 for more details.

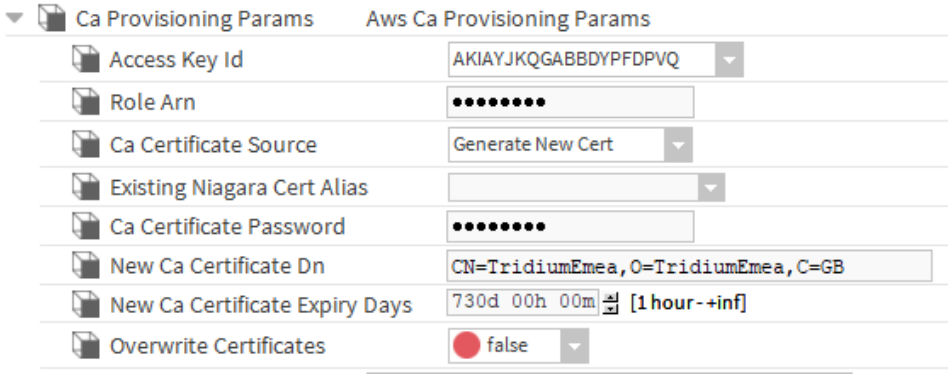


Property	Value	Description
Ca Provisioning Params	additional properties (see property table below)	Contains various properties to configure the provisioning job.
Provisioning Ca Cert Alias	read-only	Displays the alias of the CA certificate that was registered with AWS.
Mqtt Data Endpoint	read-only	Displays the retrieved endpoint to which MQTT devices can connect for this AWS account and region.
Provisioning Last Run	read-only	Displays the date and time at which the provisioning job was last executed.

Actions

Setup Just in Time Provisioning: Executes a job to register a CA with AWS.

Ca Provisioning Params folder

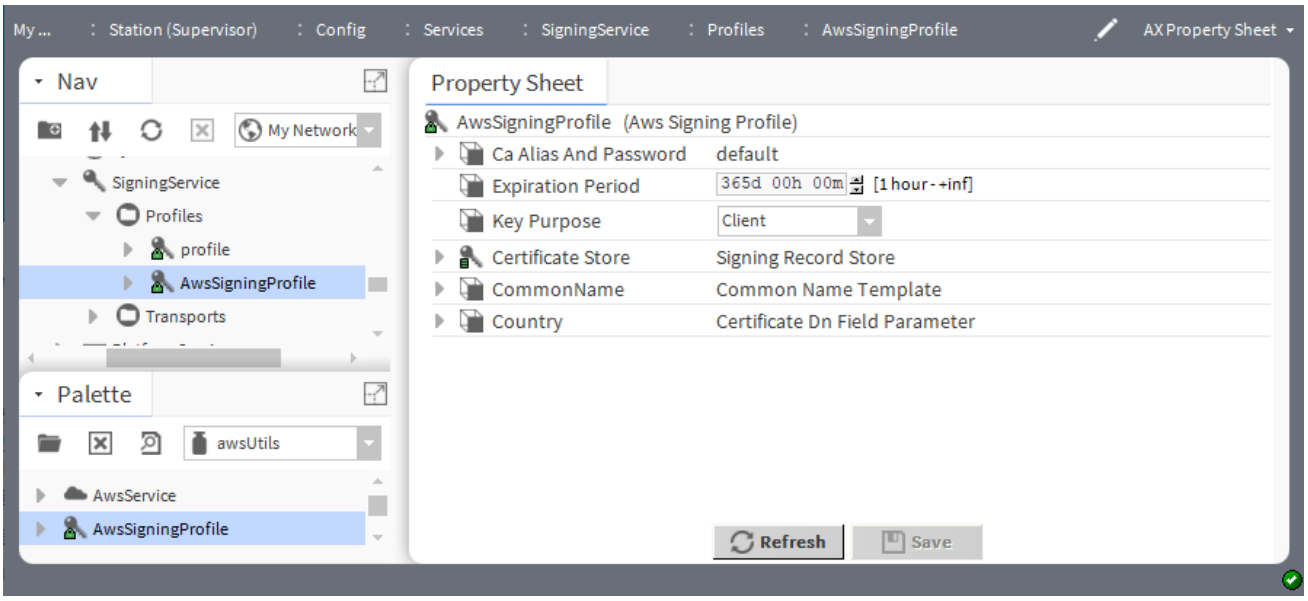


Property	Value	Description
Access Key Id	drop-down menu	Selects the access key to use for authenticating with AWS.
Role Arn	password	Specifies the ARN string for the service role that has the AWS **`AWSIoTThingsRegistration`** permission.
Ca Certificate Source	drop-down menu	Selects to generate a new CA certificate; or use an existing imported CA certificate.
Existing Niagara Cert Alias	drop-down menu	Specifies the alias of the existing imported CA certificate if selected to be used.
Ca Certificate Password	password	Specifies the password to use for the existing CA certificate, or password to use to secure the newly generated one.
New Ca Certificate Dn	string	Populates the **`Distinguished Name`** attribute list to use for a newly generated CA certificate. May optionally populate only some of the fields. CN=,O=,OU=,L=,ST=,C=
New Ca Certificate Expiry Days	days, hours, minutes (defaults to 730 days)	Defines the validity period for a newly generated CA certificate.
Overwrites Certificates	`true` or `false` (defaults to `false`)	If `true`, it will overwrite an existing CA certificate in the Key Store when generating a new CA.

Aws Signing Profile (awsUtils-AwsSigningProfile)

This component is an AWS-specific Signing Profile against which device certificates can be signed.

To use the **AWS Signing Profile**, drag it from the **`awsUtils`** palette to the **Signing Service→Profiles** folder, and populate the **`CommonName`** and **`Country`** settings to define the certificate distinguished name fields. You can further customize the profile by adding **CertificateParameters** from the **`Signing Service`** palette. See for more details.

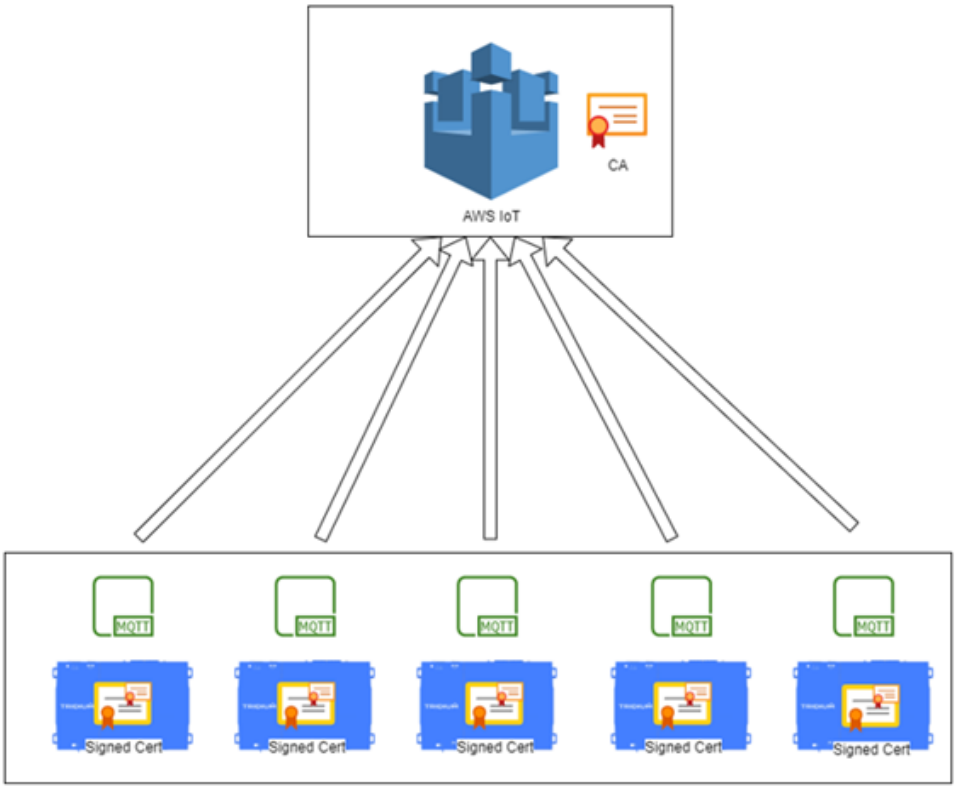


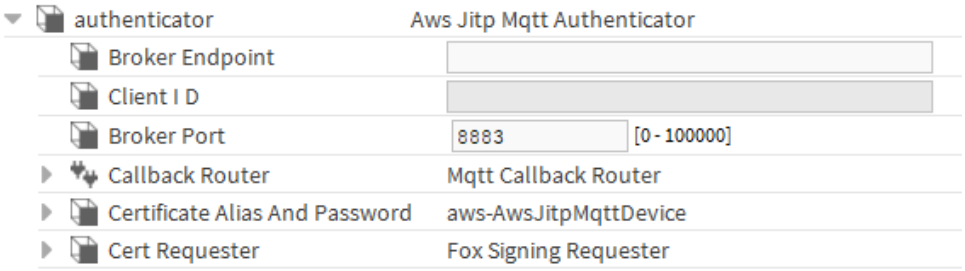
Property	Value	Description
Common Name	additional properties	Contains properties that you can use to optionally generate a unique certificate Common Name for each device.
Country	additional properties	Contains the properties to define the Certificate Dn Field Parameter.

Aws Jitp Mqtt Authenticator (abstractMqttDriver-AwsJitpMqttAuthenticator)

The **Aws Jitp Mqtt Authenticator** component connects to Amazon Web Services (AWS) utilizing the **Just In Time Provisioning** (JITP) functionality as configured in the **`awsUtils`** module. See "*Configuring Just In Time Provisioning*" in the "Niagara AWS Utils Guide" for more details.

Just In Time Provisioning allows a fleet of devices to automatically connect to AWS with auto-generated certificates as means of authentication. The major difference to the existing AWS MQTT authenticator is that the JITP authenticator does not require an AWS user to manually configure the device in AWS IoT, or to generate and sign their device certificate. This is performed in conjunction with the Signing Service, which automatically supplies signing certificates to each authenticator. In addition, certificates are also renewed without any user intervention required. For more information, see "*Signing Service*" in the "Niagara Signing Service Guide".





Property	Value	Description
Broker Endpoint	string	Defines the broker endpoint with your AWS IoT service endpoint.
Client ID	read-only	Automatically populated when the signed certificate is retrieved from the Signing Service. The value will match the Common Name of the certificate.
Broker Port	numeric value [0–100000]	Automatically set to the AWS default port 8883.
Callback Router	additional properties	Specifies **Callback Type** and **Point Callback Handler**.

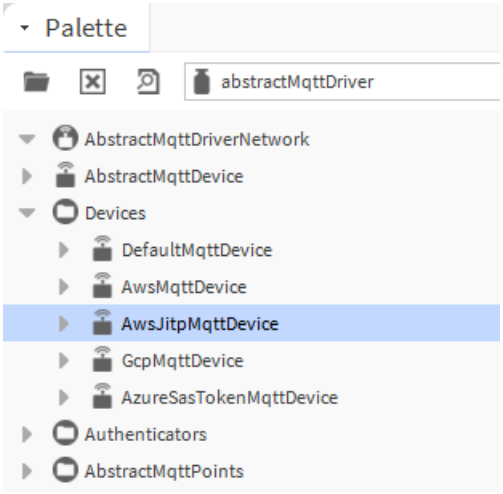
Property	Value	Description
Certificate Alias and Password	additional properties	Specifies alias and password for the certificate used to authenticate with AWS. Alias is automatically generated in the format 'aws_deviceName'
Cert Requester	additional properties	Contains components that submit a CSR to the Supervisor Signing Service and obtain the signed certificate to install in the **User Key Store**.

Automatic install

To use this authenticator, you can automatically install an MQTT device on each Niagara station in your network using a Niagara provisioning task from a Supervisor station. As the device is added to the station, it will automatically onboard with the Signing Service, obtain a signed device certificate and connect to AWS. For more information, see "Running Install AWS MQTT Device task" in the "Niagara AWS Utils Guide.

Manual install

You can also manually install a single device by dragging the **AwsJitpMqttDevice** component from the **`abstractMqttDriver`** palette.



- Populate the broker endpoint with your AWS IoT service endpoint and change the port if different from the AWS default.
- **`Certificate Alias`** will be populated automatically. We recommend that you enter a password to protect your device certificate in the Niagara **User Key Store**.
- On **`Cert Requester`**, invoke the **Onboard** action and expand this component to monitor progress. An admin user will need to approve the onboarding request in the Supervisor. For more details, see "*Signing Service*" in the "Niagara Signing Service Guide".

Glossary

Access Key	A secret credential string that identifies an AWS (Amazon Web Services) account and provides access to its services.
Access Key Id	A unique string that identifies a particular access key.
Amazon Resource Name (ARN)	A unique identifier of AWS resources. It is required to specify a resource unambiguously across Amazon Web Services (AWS).
Amazon Web Services (AWS)	Cloud computing services offered by the *Amazon Web Services* company.
AWS IoT	The Internet of Things service within *Amazon Web Services* (AWS) that offers the ability to connect to and manage remote devices via MQTT.
Just In Time Provisioning (JITP)	The ability to provision your devices within AWS IoT at the time of first connection, thereby eliminating many manual configuration tasks.

Index

A

C

D

O

P

R

S

V

W