User Guide

AWS IoT SiteWise

AWS IoT SiteWise: User Guide

Table of Contents

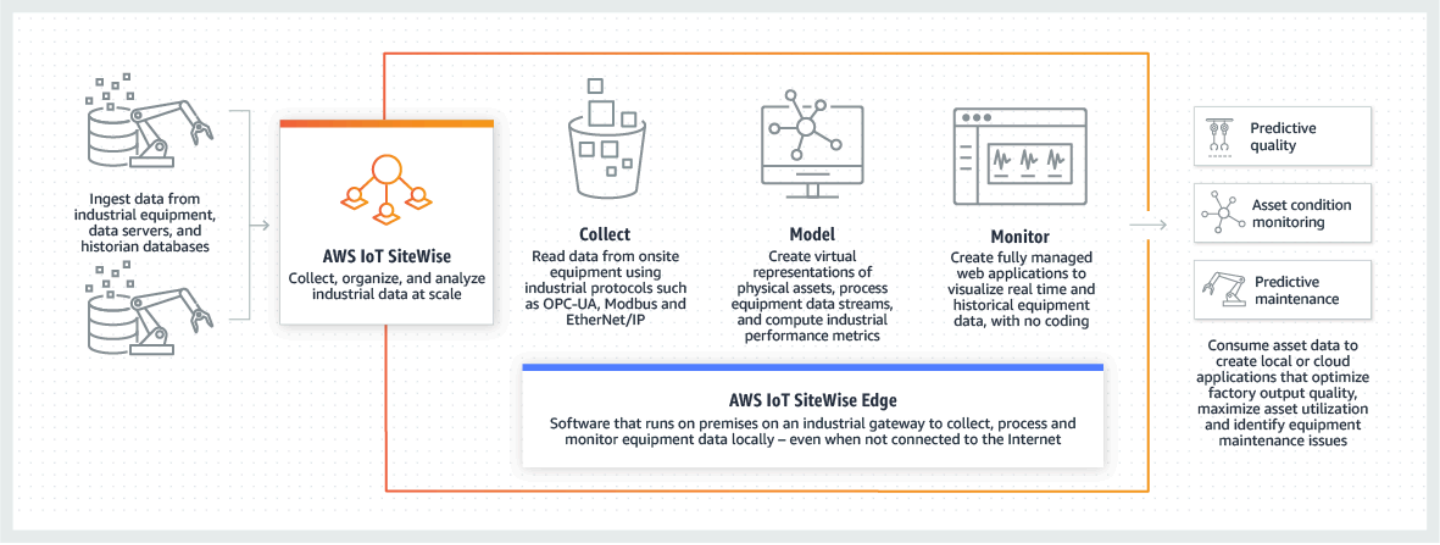
What is AWS IoT SiteWise?

AWS IoT SiteWise is a managed service that makes it easy to collect, store, organize and monitor data from industrial equipment at scale to help you make better, data-driven decisions. You can use AWS IoT SiteWise to monitor operations across facilities, quickly compute common industrial performance metrics, and create applications that analyze industrial equipment data to prevent costly equipment issues and reduce gaps in production.

AWS IoT SiteWise Monitor enables your operational users to quickly create web applications to view and analyze your industrial data in real-time. You can gain insights about your industrial operations by configuring and monitoring metrics such as *mean time between failures* and *overall equipment effectiveness (OEE)*.

AWS IoT SiteWise Edge is a component of AWS IoT SiteWise that allows collection, storage and processing of data on local devices. This is useful if you have limited access to the internet or need to keep your data private.

The following diagram shows the basic architecture of AWS IoT SiteWise:



Topics

- How AWS IoT SiteWise works
- AWS IoT SiteWise concepts
- Use cases for AWS IoT SiteWise

How AWS IoT SiteWise works

AWS IoT SiteWise offers a resource modeling framework that you can use to create representations of your industrial devices, processes, and facilities. The representations of your equipment and processes are called asset models in AWS IoT SiteWise. With asset models, you define the raw data to consume and how to process it into useful metrics. Build and visualize assets and models for your industrial operation in the AWS IoT SiteWise console. You can also configure asset models to collect and process data at the edge or in the AWS Cloud.

Topics

Ingest industrial data

Begin to use AWS IoT SiteWise by ingesting industrial data. Ingesting your data is done in one of several ways:

- **Direct ingestion from on-site servers:** Utilize protocols like OPC-UA to read data directly from on-site devices. Deploy the SiteWise Edge gateway software, compatible with AWS IoT Greengrass V2, on a wide range of platforms such as common industrial gateways or virtual servers. You can connect up to 100 OPC-UA servers to a single AWS IoT SiteWise gateway. For more information, see SiteWise Edge gateway requirements.

  Note that protocols like Modbus TCP and Ethernet/IP (EIP) are supported through our partnership with Domatica in the context of AWS IoT Greengrass V2.
- **Edge data processing with packs:** Enhance your SiteWise Edge gateway by adding packs to enable comprehensive edge capabilities. With SiteWise Edge, available on AWS IoT Greengrass V2, data processing is executed directly on-site before being securely transmitted to the AWS Cloud using an AWS IoT Greengrass stream. For more information, see Using packs.

- **Adaptive ingestion via Amazon S3 with bulk operations:** When working with large numbers of assets or asset models, use bulk operations to bulk import and export resources from Amazon S3 buckets. For more information, see Bulk operations with assets and models.
- **MQTT messages with AWS IoT Core Rules:** For devices connected to AWS IoT Core sending MQTT messages, employ the AWS IoT Core rules engine to direct those messages to AWS IoT SiteWise.If you have devices connected to AWS IoT Core sending MQTT messages, use the AWS IoT Core rules engine to route those messages to AWS IoT SiteWise. For more information, see Ingesting data using AWS IoT Core rules.
- **Event-triggered data ingestion:** Use *AWS IoT Events actions* to configure the IoT SiteWise action in AWS IoT Events to send data to AWS IoT SiteWise when events occur. For more information, see Ingesting data from AWS IoT Events.
- **AWS IoT SiteWise API:** Your applications at the Edge or in the cloud can directly send data to AWS IoT SiteWise. For more information, see Ingesting data using the AWS IoT SiteWise API

Model assets to contextualize gathered data

After ingesting data, you can use the data to create virtual representations of your assets, processes, and facilities by building models of your physical operations. An asset, representing a device or process, transmits data streams to the AWS Cloud. Assets can also signify logical device groupings. Hierarchies are formed by associating assets to mirror complex operations. These hierarchies allow assets to access data from associated child assets. Assets are created from asset models. Asset models are declarative structures that standardize asset formats. Reuse components of assets for organization and maintainability of your models. For more information, see Modeling industrial assets

With AWS IoT SiteWise, you can configure your assets to transform the incoming data into contextual metrics and transforms.

- Transforms work when receiving equipment data.
- Metrics are calculated at intervals you define.

Metrics and transforms are applicable to both individual assets or multiple assets.AWS IoT SiteWise automatically computes commonly used statistical aggregates like average, sum, and count, across various time frames relevant to your equipment data, metrics, and transforms.

Assets can be synchronized using AWS IoT TwinMaker. For more information, see

To integrate AWS IoT SiteWise and AWS IoT TwinMaker, you must have the following:

- AWS IoT SiteWise service-linked role set up in your account
- AWS IoT TwinMaker service-linked role set up in your account
- AWS IoT TwinMaker workspace with ID `IoTSiteWiseDefaultWorkspace` in your account in the Region.

To integrate by using the AWS IoT SiteWise console

When you see the **Integration with AWS IoT TwinMaker** banner in the console, choose **Grant permission**. The prerequisites are created in your account.

To integrate by using the AWS CLI

To integrate AWS IoT SiteWise and AWS IoT TwinMaker by using the AWS CLI, enter the following commands:

1. Call `CreateServiceLinkedRole` with an `AWSServiceName` of `iotsitewise.amazonaws.com`.

```
aws iam create-service-linked-role --aws-service-name iotsitewise.amazonaws.com
```

2. Call `CreateServiceLinkedRole` with an `AWSServiceName` of `iottwinmaker.amazonaws.com`.

```
aws iam create-service-linked-role --aws-service-name iottwinmaker.amazonaws.com
```

3. Call `CreateWorkspace` with an `ID` of `IoTSiteWiseDefaultWorkspace`.

```
 aws iottwinmaker create-workspace --workspace-id IoTSiteWiseDefaultWorkspace
```

.

Analyze using queries, alarms, and predictions

Analyze the date gathered with AWS IoT SiteWise by running queries and setting up alarms. You can also use Amazon Lookout to automatically detect anomalies within metrics and identify their root causes.

- Set specific alarms to alert your team when equipment or processes deviate from optimal performance, ensuring quick issue identification and resolution. For more information, see Monitoring data with alarms.
- Use the AWS IoT SiteWise API operations to query your asset properties' current values, historical values, and aggregates over specific time intervals. For more information, see Query data from AWS IoT SiteWise.
- Use anomaly detection with Amazon Lookout for Equipment to identify and visualize changes in equipment or operating conditions. With anomaly detection, you can determine preventative maintenance measures for your operations. This integration allows customers to sync data between AWS IoT SiteWise and Amazon Lookout for Equipment. For more information, see

> **Note**
>
> Anomaly detection is only available in the Regions where Amazon Lookout for Equipment is available.

You can integrate AWS IoT SiteWise with Amazon Lookout for Equipment to gain insights about your industrial equipment through anomaly detection and predictive maintenance of industrial equipment. Lookout for Equipment is a machine learning (ML) service for monitoring industrial equipment that detects abnormal equipment behavior and identifies potential failures. With Lookout for Equipment, you can implement predictive maintenance programs and identify suboptimal equipment processes. For more information about Lookout for Equipment, see What is Amazon Lookout for Equipment? in the *Amazon Lookout for Equipment User Guide*.

When you create a prediction to train an ML model to detect anomalous equipment behavior, AWS IoT SiteWise sends asset property values to Lookout for Equipment to train an ML model to detect anomalous equipment behavior. To define a prediction definition on an asset model, you specify the IAM roles needed for Lookout for Equipment to access your data and the properties to send to Lookout for Equipment and send processed data to Amazon S3. For more information, see Creating asset models.

To integrate AWS IoT SiteWise and Lookout for Equipment, you'll perform the following high-level steps:

- Add a prediction definition on an asset model that outlines what properties you want to track. The prediction definition is a reusable collection of measurements, transforms, and metrics that is used to create predictions on the assets that are based on that asset model.

- Train the prediction based on historical data that you provide.
- Schedule inference, which tells AWS IoT SiteWise how often to run a specific prediction.

Once inference is scheduled, the Lookout for Equipment model monitors the data it receives from your equipment and looks for anomalies in equipment behavior. You can view and analyze the results in SiteWise Monitor, using the AWS IoT SiteWise GET API operations, or the Lookout for Equipment console. You can also create alarms using alarm detectors from the asset model to alert you about abnormal equipment behavior.

Topics

Adding a prediction definition (console)

To begin sending data collected by AWS IoT SiteWise to Lookout for Equipment, you must add an AWS IoT SiteWise prediction definition to an asset model.

To add a prediction definition to an AWS IoT SiteWise asset model

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Models** and select the asset model to which you want to add the prediction definition.
3. Choose **Predictions**.
4. Choose **Add prediction definition**.
5. Define details about the prediction definition.

a. Enter a unique **Name** and a **Description** for your prediction definition. Choose the name thoughtfully because after you create the prediction definition, you can't change its name.

b. Create or select an **IAM permissions role** that allows AWS IoT SiteWise to share your asset data with Amazon Lookout for Equipment. The role should have the following IAM and trust policies. For help creating the role, see Creating a role using custom trust policies (console).

IAM policy

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Sid": "L4EPermissions",
            "Effect": "Allow",
            "Action": [
                "lookoutequipment:CreateDataset",
                "lookoutequipment:CreateModel",
                "lookoutequipment:CreateInferenceScheduler",
                "lookoutequipment:DescribeDataset",
                "lookoutequipment:DescribeModel",
                "lookoutequipment:DescribeInferenceScheduler",
                "lookoutequipment:ListInferenceExecutions",
                "lookoutequipment:StartDataIngestionJob",
                "lookoutequipment:StartInferenceScheduler",
                "lookoutequipment:UpdateInferenceScheduler",
                "lookoutequipment:StopInferenceScheduler"
            ],
            "Resource": [
                "arn:aws:lookoutequipment:Region:Account_ID:inference-
scheduler/IoTSiteWise_*",
                "arn:aws:lookoutequipment:Region:Account_ID:model/
IoTSiteWise_*",
                "arn:aws:lookoutequipment:Region:Account_ID:dataset/
IoTSiteWise_*"
            ]
        },
        {
            "Sid": "L4EPermissions2",
            "Effect": "Allow",
            "Action": [
```

```
                "lookoutequipment:DescribeDataIngestionJob"
            ],
            "Resource": "*"
        },
        {
            "Sid": "S3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:ListBucket",
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": ["arn:aws:s3:::iotsitewise-*"]
        },
        {
            "Sid": "IAMPermissions",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::Account_ID:role/Role_name"
        }
    ]
}
```

Trust policy

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Principal": {
                "Service": "iotsitewise.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "Account_ID"
                },
                "ArnEquals": {
```

```
                    "aws:SourceArn":
 "arn:aws:iotsitewise:Region:Account_ID:asset/*"
```

```
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lookoutequipment.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "Account_ID"
                },
                "ArnEquals": {
                    "aws:SourceArn":
 "arn:aws:lookoutequipment:Region:Account_ID:*"
                }
            }
        }
    ]
}
```

c. Choose **Next**.

6. Select data attributes (measurements, transforms, and metrics) that you want to send to Lookout for Equipment.

   a. (Optional) Select measurements.

   b. (Optional) Select transforms.

   c. (Optional) Select metrics.

   d. Choose **Next**.

7. Review your selections. To add the prediction definition to the asset model, on the summary page, choose **Add prediction definition**.

You can also **Edit** or **Delete** an existing prediction definition that has active predictions attached.

Training a prediction (console)

After you've added a prediction definition to an asset model, you can train the predictions that are on your assets.

To train a prediction in AWS IoT SiteWise

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**, and select the asset you want to monitor.
3. Choose **Predictions**.
4. Select the predictions that you want to train.
5. Under **Actions**, choose **Start training**, and do the following:
   a. Under **Prediction details**, select an IAM permissions role that allows AWS IoT SiteWise to share your asset data with Lookout for Equipment. If you need to create a new role, choose **Create a new role**.
   b. For **Training data settings**, enter a **Training data time range** to select which data to use to train the prediction.
   c. (Optional) For **Data labels**, provide an Amazon S3 bucket and prefix that holds your labeling data. For more information about labeling data, see Labeling your data in the *Amazon Lookout for Equipment User Guide*.
   d. Choose **Next**.
6. (Optional) If you want the prediction to be active as soon as it has completed training, under **Advanced settings**, select **Automatically activate the prediction after training**, and then do the following:
   a. Under **Input data**, for **Data upload frequency**, define how often data is uploaded, and for **Offset delay time**, define how much of a buffer to use.
   b. Choose **Next**.
7. Review the details of the prediction and choose **Save and start**.

Starting or stopping inference on a prediction (console)

> **Note**
>
> Lookout for Equipment charges apply to scheduled inferences with the data transferred between AWS IoT SiteWise and Lookout for Equipment. For more information, see Amazon Lookout for Equipment pricing.

If you added a prediction b"lookoutequipment:CreateDataset",ut did not choose to activate it after training, you must activate it for it to start monitoring your assets.

To start inference for a prediction

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**, and select the asset the prediction is added to.
3. Choose **Predictions**.
4. Select the predictions that you want to activate.
5. Under **Actions**, choose **Start inference**, and do the following:
   a. Under **Input data**, for **Data upload frequency**, define how often data is uploaded, and for **Offset delay time**, define how much of a buffer to use.
   b. Choose **Save and start**.

To stop inference for a prediction

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**, and select the asset the prediction is added to.
3. Choose **Predictions**.
4. Select the predictions that you want to stop.
5. Under **Actions**, choose **Stop inference**.

Adding a prediction definition (CLI)

To define a prediction definition on a new or existing asset model, you can use the AWS Command Line Interface (AWS CLI). After you define the prediction definition on the asset model, you train, and schedule inference for, a prediction on an asset in AWS IoT SiteWise to do anomaly detection with Lookout for Equipment.

Prerequisites

To complete these steps, you must have an asset model and at least one asset created. For more information, see Creating an asset model (AWS CLI) and Creating an asset (AWS CLI).

If you are new to AWS IoT SiteWise, you must call the `CreateBulkImportJob` API operation to import asset property values into AWS IoT SiteWise, which will be used to train the model. For more information, see Create a bulk import job (AWS CLI).

To add a prediction definition

1. Create a file called `asset-model-payload.json`. Follow the steps in these other sections to add your asset model's details to the file, but don't submit the request to create or update the asset model.
   - For more information about how to create an asset model, see Creating an asset model (AWS CLI)
   - For more information about how to update an existing asset model, see Updating an asset or component model (AWS CLI)
2. Add a Lookout for Equipment composite model (`assetModelCompositeModels`) to the asset model by adding the following code.
   - Replace *`Property`* with the ID of the properties that you want to include. To get those IDs, call `DescribeAssetModel`.
   - Replace *`RoleARN`* with the ARN of an IAM role that allows Lookout for Equipment to access your AWS IoT SiteWise data.

```
{
  ...
  "assetModelCompositeModels": [
```

```
    {
      "name": "L4Epredictiondefinition",
      "type": "AWS/L4E_ANOMALY",
      "properties": [
          {
            "name": "AWS/L4E_ANOMALY_RESULT",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_RESULT",
            "unit": "none",
            "type": {
              "measurement": {}
            }
          },
          {
            "name": "AWS/L4E_ANOMALY_INPUT",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_INPUT",
            "type": {
               "attribute": {
                 "defaultValue": "{\"properties\": [\"Property1\",
\"Property2\"]}"
               }
            }
          },
          {
            "name": "AWS/L4E_ANOMALY_PERMISSIONS",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_PERMISSIONS",
            "type": {
              "attribute": {
                "defaultValue": "{\"roleArn\": \"RoleARN\"}"
              }
            }
          },
          {
            "name": "AWS/L4E_ANOMALY_DATASET",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_DATASET",
            "type": {
                "attribute": {}
            }
          },
          {
            "name": "AWS/L4E_ANOMALY_MODEL",
```

```
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_MODEL",
            "type": {
              "attribute": {}
            }
          },
          {
            "name": "AWS/L4E_ANOMALY_INFERENCE",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_INFERENCE",
            "type": {
              "attribute": {}
            }
          },
          {
            "name": "AWS/L4E_ANOMALY_TRAINING_STATUS",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_TRAINING_STATUS",
            "type": {
              "attribute": {
                "defaultValue": "{}"
              }
            }
          },
          {
            "name": "AWS/L4E_ANOMALY_INFERENCE_STATUS",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_INFERENCE_STATUS",
            "type": {
              "attribute": {
                "defaultValue": "{}"
              }
            }
          }
    ]
}
```

3. Create the asset model or update the existing asset model. Do one of the following:
   - To create the asset model, run the following command:

     ```
     aws iotsitewise create-asset-model --cli-input-json file://asset-model-
     payload.json
     ```

- To update the existing asset model, run the following command. Replace *asset-model-id* with the ID of the asset model that you want to update.

```
aws iotsitewise update-asset-model \
  --asset-model-id asset-model-id \
  --cli-input-json file://asset-model-payload.json
```

After you run the command, note the `assetModelId` in the response.

Training a prediction and starting inference (CLI)

Now that the prediction definition is defined, you can train assets based on it and start inference. If you want to train your prediction but not start inference, skip to Training a prediction (CLI). To train the prediction and start inference on the asset, you'll need the `assetId` of the target resource.

To train and start inference of the prediction

1. Run the following command to find the `assetModelCompositeModelId` under `assetModelCompositeModelSummaries`. Replace *asset-model-id* with the ID of the asset model that you created in Updating an asset or component model (AWS CLI).

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. Run the following command to find the `actionDefinitionId` of the `TrainingWithInference` action. Replace *asset-model-id* with the ID used in previous step and replace *asset-model-composite-model-id* with the ID returned in the previous step.

```
aws iotsitewise describe-asset-model-composite-model \
  --asset-model-id asset-model-id \
  --asset-model-composite-model-id asset-model-composite-model-id \
```

3. Create a file called `train-start-inference-prediction.json` and add the following code, replacing the following:
   - *asset-id* with the ID of the target asset

- *action-definition-id* with the ID of the TrainingWithInference action

- *StartTime* with the start of the training data, provided in epoch seconds
- *EndTime* with the end of the training data, provided in epoch seconds

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-Id",
  "actionPayload":{
    "stringValue": "{\"l4ETrainingWithInference\":{\"trainingWithInferenceMode
\":\"START\",\"trainingPayload\":{\"exportDataStartTime\":StartTime,
\"exportDataEndTime\":EndTime},\"inferencePayload\":{\"dataDelayOffsetInMinutes
\":0,\"dataUploadFrequency\":\"PT5M\"}}}"
  }
}
```

4. Run the following command to start training and inference:

```
aws iotsitewise execute-action --cli-input-json file://train-start-inference-
prediction.json
```

Training a prediction (CLI)

Now that the prediction definition is defined, you can train assets based on it. To train the prediction on the asset, you'll need the `assetId` of the target resource.

To train the prediction

1. Run the following command to find the `assetModelCompositeModelId` under `assetModelCompositeModelSummaries`. Replace *asset-model-id* with the ID of the asset model that you created in Updating an asset or component model (AWS CLI).

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. Run the following command to find the `actionDefinitionId` of the `Training` action. Replace *`asset-model-id`* with the ID used in previous step and replace *`asset-model-composite-model-id`* with the ID returned in the previous step.

```
aws iotsitewise describe-asset-model-composite-model \
  --asset-model-id asset-model-id \
  --asset-model-composite-model-id asset-model-composite-model-id \
```

3. Create a file called `train-prediction.json` and add the following code, replacing the following:

   - *`asset-id`* with the ID of the target asset
   - *`action-definition-id`* with the ID of the training action
   - *`StartTime`* with the start of the training data, provided in epoch seconds
   - *`EndTime`* with the end of the training data, provided in epoch seconds
   - (Optional) *`BucketName`* with the name of the Amazon S3 bucket that holds your label data
   - (Optional) *`Prefix`* with the prefix associated with the Amazon S3 bucket.

   > **Note**
   >
   > Include both the bucket name and prefix or neither.

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-Id",
  "actionPayload":{ "stringValue": "{\"l4ETraining\": {\"trainingMode\":
\"START\",\"exportDataStartTime\": StartTime, \"exportDataEndTime\": EndTime,
 \"labelInputConfiguration\": {\"bucketName\": \"BucketName\", \"prefix\":
 \"Prefix\"}}}"
}
}
```

4. Run the following command to start training:

```
aws iotsitewise execute-action --cli-input-json file://train-prediction.json
```

Before you can start inference, training must be completed. To check the status of the training, do one of the following:

- From the console, navigate to the asset the prediction is on.

- From the AWS CLI, call `BatchGetAssetPropertyValue` using the `propertyId` of the `trainingStatus` property.

Starting or stopping inference on a prediction (CLI)

Once the prediction is trained, you can start inference to tell Lookout for Equipment to start monitoring your assets. To start or stop inference, you'll need the `assetId` of the target resource.

To start inference

1. Run the following command to find the `assetModelCompositeModelId` under `assetModelCompositeModelSummaries`. Replace *`asset-model-id`* with the ID of the asset model that you created in Updating an asset or component model (AWS CLI).

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. Run the following command to find the `actionDefinitionId` of the `Inference` action. Replace *`asset-model-id`* with the ID used in previous step and replace *`asset-model-composite-model-id`* with the ID returned in the previous step.

```
aws iotsitewise describe-asset-model-composite-model \
  --asset-model-id asset-model-id \
  --asset-model-composite-model-id asset-model-composite-model-id \
```

3. Create a file called `start-inference.json` and add the following code, replacing the following:
   - *`asset-id`* with the ID of the target asset
   - *`action-definition-id`* with the ID of the start inference action

- *Offset* with the amount of buffer to use

- *Frequency* with how often data is uploaded

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-Id",
  "actionPayload":{ "stringValue": "{\"l4EInference\": {\"inferenceMode\":
\"START\",\"dataDelayOffsetInMinutes\": Offset, \"dataUploadFrequency\":
 \"Frequency\"}}"
}}
```

4. Run the following command to start inference:

```
aws iotsitewise execute-action --cli-input-json file://start-inference.json
```

To stop inference

1. Run the following command to find the `assetModelCompositeModelId` under `assetModelCompositeModelSummaries`. Replace *asset-model-id* with the ID of the asset model that you created in Updating an asset or component model (AWS CLI).

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. Run the following command to find the `actionDefinitionId` of the `Inference` action. Replace *asset-model-id* with the ID used in previous step and replace *asset-model-composite-model-id* with the ID returned in the previous step.

```
aws iotsitewise describe-asset-model-composite-model \
  --asset-model-id asset-model-id \
  --asset-model-composite-model-id asset-model-composite-model-id \
```

3. Create a file called `stop-inference.json` and add the following code, replacing the following:

- *asset-id* with the ID of the target asset

- *action-definition-id* with the ID of the start inference action

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-Id",
  "actionPayload":{ "stringValue": "{\"l4EInference\":{\"inferenceMode\":\"STOP
\"}}"
}}
```

4. Run the following command to stop inference:

```
aws iotsitewise execute-action --cli-input-json file://stop-inference.json
```

.

Visualize operations

Set up SiteWise Monitor to create web applications for your operational employees. The web applications help employees to visualize your operations. Handle varied levels of access for your employees using IAM Identity Center or IAM. Configure unique logins and permissions for each employee to view specific subsets of an entire industrial operation. AWS IoT SiteWise provides an application guide for these employees to learn how to use SiteWise Monitor.

For more information on visualizing your operations, see Monitoring data with AWS IoT SiteWise Monitor

Store data

You can integrate time series storage with your industrial data lake. AWS IoT SiteWise has three storage tiers for industrial data:

- A hot storage tier that is optimized for real-time applications.
- A warm storage tier optimized for analytical workloads.
- A customer-managed cold storage tier using Amazon S3 for operational data applications with high latency tolerance.

AWS IoT SiteWise helps you manage storage cost by keeping recent data in the hot storage tier. Then, you define data retention policies to move historical data to warm or cold tier storage. For more information, see

You can configure AWS IoT SiteWise to save your data in the following storage tiers:

Hot tier

The hot storage tier is an AWS IoT SiteWise managed time series storage. Hot tier is most effective for frequently accessed data, with low write-to-read latency. Data stored in the hot tier is used by industrial applications that need quick access to the latest values of measurements in your equipment. This includes applications that visualize real-time metrics with an interactive dashboard, or applications that monitor operations and launch alarms to identify performance issues.

By default, data ingested into AWS IoT SiteWise is stored in the hot tier. You can define a retention period for the hot tier, after which AWS IoT SiteWise moves data in the hot tier to either warm or cold tier storage, based on your configuration. For best performance and cost efficiency, set your hot tier retention period to be longer than the time taken to retrieve data often. This is used for real time metrics, alarms, and monitoring scenarios. If a retention period is not set, your data is stored indefinitely in the hot tier.

Warm tier

The warm storage tier is an AWS IoT SiteWise managed tier that's effective for cost-efficient storage of historical data. It's best used to retrieve large volumes of data with medium write-to-read latency characteristics. Use the warm tier to store historical data that's needed for large workloads. For example, it's used for data retrieval for analytics, business intelligence applications (BI), reporting tools, and training of machine learning (ML) models. If you enable the cold storage tier, you can define a warm tier retention period. After the retention period ends, AWS IoT SiteWise deletes data from the warm tier.

Cold tier

The cold storage tier uses an Amazon S3 bucket to store data that's rarely used. With cold tier enabled, AWS IoT SiteWise replicates the time series, including measurements, metrics, transforms and aggregates, and asset model definitions every 6 hours. Cold tier is used to store data that tolerates high read latency for historical reports and backups.

Topics

- Configure storage settings

- Troubleshoot storage settings
- File paths and schemas of data saved in the cold tier

Configure storage settings

You can configure storage settings to opt in to service managed warm tier storage, and also to replicate data to the cold tier. To learn more about the retention period for the warm and hot tier, see Data retention impact. While configuring the storage settings, do the following:

- **Hot tier retention** — Set a retention period for how long your data is stored in the hot tier before it's deleted, and moved to the service managed warm tier storage or cold tier storage based on your storage settings. AWS IoT SiteWise will delete any data in the hot tier that existed before the retention period ends. If you don't set a retention period, your data is stored indefinitely in the hot tier.
- **Warm tier retention** — Set a retention period for how long your data is stored in the warm tier before it's deleted from AWS IoT SiteWise storage and moved to the customer managed cold tier storage. AWS IoT SiteWise deletes any data from the warm tier that existed before the retention period ends. If a retention period is not set, your data is stored indefinitely in the warm tier.

> **Note**
>
> To improve query performance, set a hot tier retention period with warm tier storage.

Impact of data retention in hot and warm tier storage

- When you decrease the retention period of the hot tier storage, data is permanently moved from the hot tier to the warm or cold tier. When you decrease the retention period of the warm tier, data is moved to the cold tier, and permanently deleted from the warm tier.
- When you increase the retention period of the hot or warm tier storage, the change affects data that's sent to AWS IoT SiteWise from then on. AWS IoT SiteWise does not retrieve data from the warm or cold storage to populate the hot tier. For example, if the retention period of the hot tier storage is initially set for 30 days and then increased to 60 days, it takes 30 days for the hot tier storage to contain 60 days worth of data.

Topics

Configure storage settings for warm tier (console)

The following procedure shows you how to configure the storage settings to replicate data to the warm tier in the AWS IoT SiteWise console.

To configure storage settings in the console

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, under **Settings**, choose **Storage**.
3. In the upper-right corner, choose **Edit**.
4. On the **Edit storage** page, do the following:
5. For **Hot tier settings**, do the following:
   - If you want to set a retention period for how long your data is stored in the hot tier before it's deleted, and moved to the service managed warm tier storage, choose **Enable retention period**.
   - To configure a retention period, enter a whole number and choose a unit. The retention period must be greater than or equal to 30 days.

   AWS IoT SiteWise deletes any data in the hot tier that's older than the retention period. If you don't set a retention period, your data is stored indefinitely.
6. (Recommended) For **Warm tier settings**, do the following:
   - To opt in to warm tier storage, select **I confirm to the opt-in of warm tier storage** to opt in for the warm tier storage.
   - (Optional) To configure a retention period, enter a whole number and choose a unit. The retention period must be greater than or equal to 365 days.

AWS IoT SiteWise deletes data in the warm tier that existed earlier than the retention period. If you don't set a retention period, your data is stored indefinitely.

Note

- When you opt in for warm tier, the configuration displays once only.
- To set hot tier retention, you must have either warm or cold tier storage. For cost efficiency and historical data retrieval, AWS IoT SiteWise recommends that you store long term data in the warm tier.
- To set warm tier retention, you must have cold tier storage.

7. Choose **Save** to save your storage settings.

In the **AWS IoT SiteWise storage** section, the **Warm tier storage** is in one of these states:

- **Enabled** – If your data existed before the hot tier retention period, AWS IoT SiteWise moves the data to the warm tier."
- **Disabled** – The warm tier storage is disabled.

Configure storage settings for warm tier (AWS CLI)

You can configure storage settings to move data to the warm tier by using the AWS CLI and the following commands.

To prevent overriding the existing configuration, retrieve the current storage configuration information by running the following command:

```
aws iotsitewise describe-storage-configuration
```

Example response without existing cold tier configuration

```
{
        "storageType": "SITEWISE_DEFAULT_STORAGE",
        "disassociatedDataStorage": "ENABLED",
        "configurationStatus": {
```

```
            "state": "ACTIVE"
        },
        "lastUpdateDate": "2021-10-14T15:53:35-07:00",
        "warmTier": "DISABLED"
}
```

Example response with existing cold tier configuration

```
{
    "storageType": "MULTI_LAYER_STORAGE",
        "multiLayerStorage": {
          "customerManagedS3Storage": {
          "s3ResourceArn": "arn:aws:s3:::bucket-name/prefix/",
          "roleArn": "arn:aws:iam::aws-account-id:role/role-name"
          }
        },
    "disassociatedDataStorage": "ENABLED",
    "retentionPeriod": {
    "numberOfDays": retention-in-days
    },
     "configurationStatus": {
     "state": "ACTIVE"
    },
    "lastUpdateDate": "2023-10-25T15:59:46-07:00",
    "warmTier": "DISABLED"
}
```

```
            "numberOfDays": hot-tier-retention-in-days
        }

}
```

hot-tier-retention-in-days must be a whole number greater than or equal to 30 days.

Example response

```
{
        "storageType": "SITEWISE_DEFAULT_STORAGE",
        "configurationStatus": {
        "state": "UPDATE_IN_PROGRESS"
        }
}
```

If you have cold tier storage enabled, see Configure storage settings with AWS CLI and existing cold tier.

Configure storage settings with AWS CLI and existing cold tier

Configure storage settings using AWS CLI with existing cold tier storage

- Run the following command to configure the storage settings. Replace *file-name* with the name of the file that contains the AWS IoT SiteWise storage configuration.

  ```
  aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
  ```

  Example AWS IoT SiteWise storage configuration

  - Replace *bucket-name* with your Amazon S3 bucket name.
  - Replace *prefix* with your Amazon S3 prefix.
  - Replace *aws-account-id* with your AWS account ID.
  - Replace *role-name* with the name of the Amazon S3 access role that allows AWS IoT SiteWise to send data to Amazon S3.
  - Replace *hot-tier-retention-in-days* with a whole number greater than or equal to 30 days.
  - Replace *warm-tier-retention-in-days* with a whole number greater than or equal to

> **Note**
>
> AWS IoT SiteWise will delete any data in the warm tier that's older than the retention period of the cold tier. If you don't set a retention period, your data is stored indefinitely.

```
{
      "storageType": "MULTI_LAYER_STORAGE",
        "multiLayerStorage": {
          "customerManagedS3Storage": {
              "s3ResourceArn": "arn:aws:s3:::bucket-name/prefix/",
              "roleArn": "arn:aws:iam::aws-account-id:role/role-name"
              }
          },
    "disassociatedDataStorage": "ENABLED",
    "retentionPeriod": {
      "numberOfDays": hot-tier-retention-in-days
    },
    "warmTier": "ENABLED",
    "warmTierRetentionPeriod": {
      "numberOfDays": warm-tier-retention-in-days
    }
}
```

Example response

```
{
      "storageType": "MULTI_LAYER_STORAGE",
      "configurationStatus": {
        "state": "UPDATE_IN_PROGRESS"
       }
}
```

Configure storage settings for cold tier (console)

The following procedure shows you how to configure the storage settings to replicate data to the cold tier in the AWS IoT SiteWise console.

To configure storage settings in the console

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, under **Settings**, choose **Storage**.
3. In the upper-right corner, choose **Edit**.
4. On the **Edit storage** page, do the following:
   a. For **Storage settings**, choose **Enable cold tier storage**. The cold tier storage is disabled by default.
   b. For **S3 bucket location**, enter the name of an existing Amazon S3 bucket and a prefix.

      Note

      - Amazon S3 uses the prefix as a folder name in the Amazon S3 bucket. The prefix must have 1-255 characters and end with a forward slash (/). Your AWS IoT SiteWise data is saved in this folder.
      - If you don't have an Amazon S3 bucket, choose **View**, and then create one in the Amazon S3 console. For more information, see Create your first S3 bucket in the *Amazon S3 User Guide*.

   c. For **S3 access role**, do one of the following:
      - Choose **Create a role from an AWS managed template**, AWS automatically creates an IAM role that allows AWS IoT SiteWise to send data to Amazon S3.
      - Choose **Use an existing role**, and then choose the role that you created from the list.

      Note

      - You must use the same Amazon S3 bucket name for the **S3 bucket location** that you used in the previous step and in your IAM policy.
      - Make sure that your role has the permissions shown in the following example. **Example permissions policy:**

```
{
      "Version": "2012-10-17",
      "Statement": [
```

```
            {
                "Effect": "Allow",
                "Action": [
                    "s3:PutObject",
                    "s3:GetObject",
                    "s3:DeleteObject",
                    "s3:GetBucketLocation",
                    "s3:ListBucket"
                ],
                "Resource": [
                    "arn:aws:s3:::bucket-name",
                    "arn:aws:s3:::bucket-name/*"
                ]
            }
        ]
    }
```

Replace *bucket-name* with the name of your Amazon S3 bucket.

d. To setup hot tier, see Step 5 in Configure storage settings for warm tier (console).

e. (Optional) For **AWS IoT Analytics integration**, do the following.

   i. If you want to use AWS IoT Analytics to query your data, choose **Enabled AWS IoT Analytics data store**.

   ii. AWS IoT SiteWise generates a name for your data store or you can enter a different name.

   AWS IoT SiteWise automatically creates a data store in AWS IoT Analytics to save your data. To query the data, you can use AWS IoT Analytics to create datasets. For more information, see Working with AWS IoT SiteWise data in the *AWS IoT Analytics User Guide*.

f. Choose **Save**.

In the **AWS IoT SiteWise storage** section, the **Cold tier storage** can be one of the following values:

- **Enabled** – AWS IoT SiteWise replicates your data to the specified Amazon S3 bucket.
- **Enabling** – AWS IoT SiteWise is processing your request to enable the cold tier storage. This process can take several minutes to complete.

- **Enable_Failed** – AWS IoT SiteWise couldn't process your request to enable the cold tier storage. If you enabled AWS IoT SiteWise to send logs to Amazon CloudWatch Logs, you can use these logs to troubleshoot issues. For more information, see Monitoring with Amazon CloudWatch Logs.
- **Disabled** – The cold tier storage is disabled.

Configure storage settings for cold tier (AWS CLI)

The following procedure shows you how to configure the storage settings to replicate data to the cold tier using AWS CLI.

To configure storage settings using AWS CLI

1. To export data to an Amazon S3 bucket in your account, run the following command to configure the storage settings. Replace *file-name* with the name of the file that contains the AWS IoT SiteWise storage configuration.

```
aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
```

Example AWS IoT SiteWise storage configuration

- Replace *bucket-name* with your Amazon S3 bucket name.
- Replace *prefix* with your Amazon S3 prefix.
- Replace *aws-account-id* with your AWS account ID.
- Replace *role-name* with the name of the Amazon S3 access role that allows AWS IoT SiteWise to send data to Amazon S3.
- Replace *retention-in-days* with a whole number than is greater than or equal to 30 days.

```
{
      "storageType": "MULTI_LAYER_STORAGE",
      "multiLayerStorage": {
          "customerManagedS3Storage": {
              "s3ResourceArn": "arn:aws:s3:::bucket-name/prefix/",
              "roleArn": "arn:aws:iam::aws-account-id:role/role-name"
          }
      },
```

```
    "retentionPeriod": {
        "numberOfDays": retention-in-days,
        "unlimited": false
    }
}
```

Note

- You must use the same Amazon S3 bucket name in the AWS IoT SiteWise storage configuration and IAM policy.
- Make sure that your role has the permissions shown in the following example.
  Example permissions policy:

```
{
      "Version": "2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": [
                  "s3:PutObject",
                  "s3:GetObject",
                  "s3:DeleteObject",
                  "s3:GetBucketLocation",
                  "s3:ListBucket"
              ],
              "Resource": [
                  "arn:aws:s3:::bucket-name",
                  "arn:aws:s3:::bucket-name/*"
              ]
          }
      ]
  }
```

  Replace *bucket-name* with the name of your Amazon S3 bucket.

Example response

```
{
```

```
    "storageType": "MULTI_LAYER_STORAGE",
    "retentionPeriod": {
        "numberOfDays": 100,
        "unlimited": false
    },
    "configurationStatus": {
        "state": "UPDATE_IN_PROGRESS"
    }
}
```

> **Note**
>
> It can take a few minutes for AWS IoT SiteWise to update the storage configuration.

2. To retrieve the storage configuration information, run the following command.

```
aws iotsitewise describe-storage-configuration
```

Example response

```
{
      "storageType": "MULTI_LAYER_STORAGE",
      "multiLayerStorage": {
          "customerManagedS3Storage": {
              "s3ResourceArn": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/torque/",
              "roleArn": "arn:aws:iam::123456789012:role/SWAccessS3Role"
          }
      },
      "retentionPeriod": {
          "numberOfDays": 100,
          "unlimited": false
      },
      "configurationStatus": {
          "state": "ACTIVE"
      },
      "lastUpdateDate": "2021-03-30T15:54:14-07:00"
  }
```

3. To stop exporting data to the Amazon S3 bucket, run the following command to configure storage settings.

```
aws iotsitewise put-storage-configuration --storage-type SITEWISE_DEFAULT_STORAGE
```

> **Note**
>
> By default, your data is only stored in the hot tier of AWS IoT SiteWise.

Example response

```
{
      "storageType": "SITEWISE_DEFAULT_STORAGE",
      "configurationStatus": {
          "state": "UPDATE_IN_PROGRESS"
      }
  }
```

4. To retrieve the storage configuration information, run the following command.

```
aws iotsitewise describe-storage-configuration
```

Example response

```
{
      "storageType": "SITEWISE_DEFAULT_STORAGE",
      "configurationStatus": {
          "state": "ACTIVE"
      },
      "lastUpdateDate": "2021-03-30T15:57:14-07:00"
  }
```

(Optional) Create an AWS IoT Analytics data store (AWS CLI)

An AWS IoT Analytics data store is a scalable and queryable repository that receives and stores data. You can use the AWS IoT SiteWise console or AWS IoT Analytics APIs to create an AWS IoT Analytics data store to save your AWS IoT SiteWise data. To query the data, you create datasets by using AWS IoT Analytics. For more information, see Working with AWS IoT SiteWise data in the *AWS IoT Analytics User Guide*.

The following steps use AWS CLI to create a data store in AWS IoT Analytics.

To create a data store, run the following command. Replace *file-name* with the name of the file that contains the data store configuration.

```
aws iotanalytics create-datastore --cli-input-json file://file-name.json
```

Note

- You must specify the name of an existing Amazon S3 bucket. If you don't have an Amazon S3 bucket, create one first. For more information, see Create your first S3 bucket in *Amazon S3 User Guide*.

- You must use the same Amazon S3 bucket name in the AWS IoT SiteWise storage configuration, IAM policy, and AWS IoT Analytics data store configuration.

Example AWS IoT Analytics data store configuration

Replace *data-store-name* and *s3-bucket-name* with your AWS IoT Analytics data store name and Amazon S3 bucket name.

```
{
      "datastoreName": "data-store-name",
      "datastoreStorage": {
          "iotSiteWiseMultiLayerStorage": {
              "customerManagedS3Storage": {
                  "bucket": "s3-bucket-name"
              }
          }
      },
      "retentionPeriod": {
          "numberOfDays": 90
```

```
      "datastoreArn": "arn:aws:iotanalytics:us-west-2:123456789012:datastore/
datastore_IoTSiteWise_demo",
      "retentionPeriod": {
          "numberOfDays": 90,
          "unlimited": false
      }
  }
```

Troubleshoot storage settings

Use the following information to troubleshoot and resolve issues with the storage configuration.

Issues

- Error: Bucket doesn't exist
- Error: Access denied to Amazon S3 path
- Error: Role ARN can't be assumed
- Error: Failed to access cross-Region Amazon S3 bucket

Error: Bucket doesn't exist

Solution: AWS IoT SiteWise couldn't find your Amazon S3 bucket. Make sure you enter the name of an existing Amazon S3 bucket in the current Region.

Error: Access denied to Amazon S3 path

Solution: AWS IoT SiteWise couldn't access your Amazon S3 bucket. Do the following:

- Make sure that you use the same Amazon S3 bucket that you specified in the IAM policy.
- Make sure that your role has the permissions shown in the following example.

Example permissions policy

```
{
      "Version": "2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": [
```

```
                "s3:PutObject",
```

```
                "s3:GetObject",
                "s3:DeleteObject",
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

Replace *bucket-name* with the name of your Amazon S3 bucket.

Error: Role ARN can't be assumed

Solution: AWS IoT SiteWise couldn't assume the IAM role on your behalf. Make sure that your role trusts the following service: `iotsitewise.amazonaws.com`. For more information, see I can't assume a role see *IAM User Guide*.

Error: Failed to access cross-Region Amazon S3 bucket

Solution: The Amazon S3 bucket that you specified is in a different AWS Region. Make sure that your Amazon S3 bucket and AWS IoT SiteWise assets are in the same Region.

File paths and schemas of data saved in the cold tier

AWS IoT SiteWise stores your data in the cold tier by replicating time series, including measurements, metrics, transforms and aggregates, and also asset and asset model definitions. The following describes the file paths and schemas of data that is sent to the cold tier.

Topics

- Storage data index files

Equipment data (measurements)

AWS IoT SiteWise exports equipment data (measurements) to the cold tier once every six hours. Raw data is saved in the cold tier in the Apache AVRO (`.avro`) format.

File path

AWS IoT SiteWise stores equipment data (measurements) in the cold tier using the following template.

```
{keyPrefix}/raw/startYear={startYear}/startMonth={startMonth}/startDay={startDay}/
```

```
seriesBucket={seriesBucket}/raw_{timeseriesId}_{startTimestamp}_{quality}.avro
```

Every file path to raw data in Amazon S3 contains the following components.

`keyPrefix`	The Amazon S3 prefix that you specified in the AWS IoT SiteWise storage configuration. Amazon S3 uses the prefix as a folder name in the bucket.
`raw`	The folder that stores time series data from equipment (measurements). The `raw` folder is saved in the prefix folder.
`seriesBucket`	A hexadecimal number between 00 and ff. This number is derived from `timeSeriesId` . This partition is used to increase throughput when AWS IoT SiteWise writes to the cold tier. When you use Amazon Athena to run queries, you can use the partition for fine-grain partitioning to improve query performance. `seriesBucket` and `timeSeriesBucket` in the asset metadata are the same number.

`startYear`	The year of the exclusive start time associated with the time series data.
`startMonth`	The month of the exclusive start time associated with the time series data.
`startDay`	The day of the month of the exclusive start time associated with the time series data.
`fileName`	The file name uses the underscore (_) character as a delimiter to separate the following: • The `raw` prefix. • The `timeSeriesId` value. • The epoch timestamp of the exclusive start time associated with the time series data. • The quality of the data. Valid values: `GOOD`, `BAD`, and `UNCERTAIN`. For more information, see AssetPropertyValue in the *AWS IoT SiteWise API Reference*. The file is saved in the `.avro` format by using the Snappy compression.

Path component	Description

Example file path to raw data in the cold tier

```
keyPrefix/raw/startYear=2021/startMonth=1/startDay=2/seriesBucket=a2/
raw_7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a_1609577700_GOOD.avro
```

Fields

The schema of raw data that is exported to the cold tier contains the following fields.

`seriesId`	`string`	N/A	The ID that identifies the time series data from equipment (measurements). You can use this field to join raw data and asset metadata in queries.
`timeInSeconds`	`long`	N/A	The timestamp date, in seconds, in the Unix epoch format. Fractional nanosecond data is provided by `offsetInNanos`.
`offsetInNanos`	`long`	N/A	The nanosecond offset from `timeInSeconds`.
`quality`	`string`	N/A	The quality of the time series value.
`doubleValue`	`double` or `null`	`null`	Time series data of type double (floating point number).
`stringValue`	`string` or `null`	`null`	Time series data of type string (sequence of characters).
`integerValue`	`int` or `null`	`null`	Time series data of type integer (whole number).

booleanValue	boolean or null	null	Time series data of type Boolean (true or false).
jsonValue	string or null	null	Time series data of type JSON (complex data types stored as a string).
recordVersion	long or null	null	The version number for the record. You can use the version number to select the latest record. Newer records have larger version numbers.

Field name	Supported types	Default type	Description

Example raw data in the cold tier


```
{"seriesId":"e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeInSeconds":1625675887,"offsetInNanos":0,"quality":"GOOD","doubleValue":
{"double":0.75},"stringValue":null,"integerValue":null,"booleanValue":null,"jsonValue":null,"re
  {"seriesId":"e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeInSeconds":1625675889,"offsetInNanos":0,"quality":"GOOD","doubleValue":
{"double":0.69},"stringValue":null,"integerValue":null,"booleanValue":null,"jsonValue":null,"re
  {"seriesId":"e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeInSeconds":1625675890,"offsetInNanos":0,"quality":"GOOD","doubleValue":
{"double":0.66},"stringValue":null,"integerValue":null,"booleanValue":null,"jsonValue":null,"re
  {"seriesId":"e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeInSeconds":1625675891,"offsetInNanos":0,"quality":"GOOD","doubleValue":
{"double":0.92},"stringValue":null,"integerValue":null,"booleanValue":null,"jsonValue":null,"re
```

Metrics, transforms, and aggregates

AWS IoT SiteWise exports metrics, transforms, and aggregates to the cold tier once every six hours. Metrics, transforms, and aggregates are saved in the cold tier in the Apache AVRO (`.avro`) format.

File path

AWS IoT SiteWise stores metrics, transforms, and aggregates in the cold tier using the following template.

```
{keyPrefix}/agg/startYear={startYear}/startMonth={startMonth}/startDay={startDay}/
```

```
seriesBucket={seriesBucket}/agg_{timeseriesId}_{startTimestamp}_{quality}.avro
```

Every file path to metrics, transforms, and aggregates in Amazon S3 contains the following components.

`keyPrefix`	The Amazon S3 prefix that you specified in the AWS IoT SiteWise storage configuration. Amazon S3 uses the prefix as a folder name in the bucket.
`agg`	The folder that stores time series data from metrics. The `agg` folder is saved in the prefix folder.
`seriesBucket`	A hexadecimal number between 00 and ff. This number is derived from `timeSeriesId` . This partition is used to increase throughput when AWS IoT SiteWise writes to the cold tier. When you use Amazon Athena to run queries, you can use the partition for fine-grain partitioning to improve query performance. `seriesBucket` and `timeSeriesBucket` in the asset metadata are the same number.

`startYear`	The year of the exclusive start time associated with the time series data.
`startMonth`	The month of the exclusive start time associated with the time series data.
`startDay`	The day of the month of the exclusive start time associated with the time series data.
`fileName`	The file name uses the underscore (_) character as a delimiter to separate the following: • The `raw` prefix. • The `timeSeriesId` value. • The epoch timestamp of the exclusive start time associated with the time series data. • The quality of the data. Valid values: `GOOD`, `BAD`, and `UNCERTAIN`. For more information, see AssetPropertyValue in the *AWS IoT SiteWise API Reference*. The file is saved in the `.avro` format by using the Snappy compression.

Path component	Description

Example file path to metrics in the cold tier

```
keyPrefix/agg/startYear=2021/startMonth=1/startDay=2/seriesBucket=a2/
agg_7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a_1609577700_GOOD.avro
```

Fields

The schema of metrics, transforms, and aggregates that are exported to the cold tier contains the following fields.

`seriesId`	`string`	N/A	The ID that identifies the time series data from equipment, metrics, or transforms. You can use this field to join raw data and asset metadata in queries.
`timeInSeconds`	`long`	N/A	The timestamp date, in seconds, in the Unix epoch format. Fractional nanosecond data is provided by `offsetInNanos`.
`offsetInNanos`	`long`	N/A	The nanosecond offset from `timeInSeconds`.
`quality`	`string`	N/A	The quality by which to filter asset data.
`resolution`	`string`	N/A	The time interval over which to aggregate data.
`count`	`double` or `null`	`null`	The total number of data points for the given variables over the current time interval.
`average`	`double` or `null`	`null`	The mean of the given variables

			' values over the current time interval.
`min`	`double` or `null`	`null`	The minimum of the given variables ' values over the current time interval.
`max`	`boolean` or `null`	`null`	The maximum of the given variables ' values over the current time interval.
`sum`	`string` or `null`	`null`	The sum of the given variables' values over the current time interval.
`recordVersion`	`long` or `null`	`null`	The version number for the record. You can use the version number to select the latest record. Newer records have larger version numbers.

Field name	Supported types	Default type	Description

Example Metric data in the cold tier

```
{"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637334060,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"double":16.0},"min":{"double":1.0},"max":{"double":31.0},"sum":
{"double":496.0},"recordVersion":null}
  {"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637334120,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"double":46.0},"min":{"double":32.0},"max":{"double":60.0},"sum":
```

```
  {"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637334540,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"double":16.0},"min":{"double":1.0},"max":{"double":31.0},"sum":
{"double":496.0},"recordVersion":null}
  {"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637334600,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"double":46.0},"min":{"double":32.0},"max":{"double":60.0},"sum":
{"double":1334.0},"recordVersion":null}
  {"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637335020,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"double":16.0},"min":{"double":1.0},"max":{"double":31.0},"sum":
{"double":496.0},"recordVersion":null}
```


Asset metadata

When you enable AWS IoT SiteWise to export data to the cold tier for the first time, asset metadata is exported to the cold tier. After the initial configuration, AWS IoT SiteWise exports asset metadata to the tier only when you change asset model definitions or asset definitions. Asset metadata is saved in the cold tier in the newline delimited JSON (`.ndjson`) format.

File path

AWS IoT SiteWise stores asset metadata in the cold tier using the following template.

```
{keyPrefix}/asset_metadata/asset_{assetId}.ndjson
```

Every file path to asset metadata in the cold tier contains the following components.

`keyPrefix`	The Amazon S3 prefix that you specified in the AWS IoT SiteWises storage configuration. Amazon S3 uses the prefix as a folder name in the bucket.
`asset_metadata`	The folder that stores asset metadata. The `asset_metadata` folder is saved in the

`fileName`	The file name uses the underscore (_) character as a delimiter to separate the following:
	• The `asset` prefix. • The `assetId` value. The file is saved in the `.ndjson` format.

Path component	Description

Example file path to asset metadata in the colder tier

```
keyPrefix/asset_metadata/asset_35901915-d476-4dca-8637-d9ed4df939ed.ndjson
```

Fields

The schema of asset metadata that is exported to the cold tier contains the following fields.

`assetId`	The ID of the asset.
`assetName`	The name of the asset.
`assetExternalId`	The external ID of the asset.
`assetModelId`	The ID of the asset model used to create this asset.
`assetModelName`	The name of the asset model.
`assetModelExternalId`	The external ID of the asset model.
`assetPropertyId`	The ID of the asset property.
`assetPropertyName`	The name of the asset property.
`assetPropertyExternalId`	The external ID of the asset property.

`assetPropertyDataType`	The data type of the asset property.
`assetPropertyUnit`	The unit of the asset property (for example, `Newtons` and `RPM`).
`assetPropertyAlias`	The alias that identifies the asset property, such as an OPC-UA server data stream path (for example, `/company/windfarm/3/turbine/7/temperature`).
`timeSeriesId`	The ID that identifies the time series data from equipment, metrics, or transforms. You can use this field to join raw data and asset metadata in queries.
`timeSeriesBucket`	A hexadecimal number between 00 and ff. This number is derived from `timeSeriesId` . This partition is used to increase throughput when AWS IoT SiteWise writes to the cold tier. When you use Amazon Athena to run queries, you can use the partition for fine-grain partitioning to improve query performance. `timeSeriesBucket` and `seriesBucket` in the file path to raw data are the same number.
`assetCompositeModelId`	The ID of the composite model.
`assetCompositeModelExternalId`	The external ID of the composite model.
`assetCompositeModelDescription`	The description of the composite model.
`assetCompositeModelName`	The name of the composite model.
`assetCompositeModelType`	The type of the composite model. For alarm composite models, this type is `AWS/ALARM` .

assetCreationDate	The date the asset was created, in Unix epoch time.
assetLastUpdateDate	The date the asset was last updated, in Unix epoch time.
assetStatusErrorCode	The error code.
assetStatusErrorMessage	The error message.
assetStatusState	The current status of the asset.

Field name	Description

Example asset metadata in the cold tier

```
  {"assetId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d","assetExternalId":null,"assetName":"Wind Turbine Asset
 2","assetModelId":"ec1d924f-f07d-444f-b072-
e2994c165d35","assetModelExternalId":null,"assetModelName":"Wind
 Turbine Asset Model","assetPropertyId":"95e63da7-d34e-43e1-
bc6f-1b490154b07a","assetPropertyExternalId":null,"assetPropertyName":"Temperature","assetPrope
Washington/Seattle/WT2/temp","timeSeriesId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeSeriesBucket":"f6","assetArn":null,"assetCompositeModelDescription":nul
  {"assetId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d","assetExternalId":null,"assetName":"Wind Turbine Asset
 2","assetModelId":"ec1d924f-f07d-444f-b072-
e2994c165d35","assetModelExternalId":null,"assetModelName":"Wind Turbine Asset
 Model","assetPropertyId":"c706d54d-4c11-42dc-9a01-63662fc697b4","assetPropertyExternalId":null
```

```
e6db-40fa-9845-ed0dddd4c77d_8cf1162f-dead-4fbe-b468-

c8e24cde9f50","timeSeriesBucket":"d7","assetArn":null,"assetCompositeModelDescription":null,"as

 {"assetId":"3a5f2a22-3b37-4332-9c1c-404ea1d73fab","assetExternalId":null,"assetName":"BatchAss

ebc75e75e827","assetModelExternalId":null,"assetModelName":"FlashTestAssetModelDouble","assetPr

b410-

ab401a9176ed","assetPropertyExternalId":null,"assetPropertyName":"measurementProperty","assetPr

ae89-

ff316f5ff8aa","timeSeriesBucket":"af","assetArn":null,"assetCompositeModelDescription":null,"as
```


Asset hierarchy metadata

When you enable AWS IoT SiteWise to save data the in cold tier for the first time, asset hierarchy metadata is exported to the cold tier. After the initial configuration, AWS IoT SiteWise exports asset hierarchy metadata to the cold tier only when you make changes to asset model or asset definitions. Asset hierarchy metadata is saved in the cold tier in the newline delimited JSON (`.ndjson`) format.

An external identifier for the hierarchy, target asset, or source asset is retrieved by calling the DescribeAsset API.

File path

AWS IoT SiteWise stores asset hierarchy metadata in the cold tier using the following template.

```
{keyPrefix}/asset_hierarchy_metadata/{parentAssetId}_{hierarchyId}.ndjson
```

Every file path to asset hierarchy metadata in the cold tier contains the following components.

`keyPrefix`	The Amazon S3 prefix that you specified in the AWS IoT SiteWise storage configuration. Amazon S3 uses the prefix as a folder name in the bucket.

`asset_hierarchy_metadata`	The folder that stores asset hierarchy metadata. The `asset_hierarchy_metadata` folder is saved in the prefix folder.

`fileName`	The file name uses the underscore (_) character as a delimiter to separate the following: • The `parentAssetId` value. • The `hierarchyId` value. The file is saved in the `.ndjson` format.

Path component	Description

Example file path to asset hierarchy metadata in the cold tier

```
keyPrefix/asset_hierarchy_metadata/35901915-d476-4dca-8637-
d9ed4df939ed_c5b3ced8-589a-48c7-9998-cdccfc9747a0.ndjson
```

Fields

The schema of asset hierarchy metadata that is exported to the cold tier contains the following fields.

`sourceAssetId`	The ID of the source asset in this asset relationship.
`targetAssetId`	The ID of the target asset in this asset relationship.
`hierarchyId`	The ID of the hierarchy.
`associationType`	The association type of this asset relationship.

	The value must be CHILD. The target asset is a child asset of the source asset.

Field name	Description

Example asset hierarchy metadata in the cold tier

```
{"sourceAssetId":"80388e72-2284-44fb-9c89-
bfbaf0dfedd2","targetAssetId":"2b866c25-0c74-4750-bdf5-
b73683c8a2a2","hierarchyId":"bbed9f59-0412-4585-
a61d-6044db526aee","associationType":"CHILD"}
  {"sourceAssetId":"80388e72-2284-44fb-9c89-
bfbaf0dfedd2","targetAssetId":"6b51246e-984d-460d-
bc0b-470ea47d1e31","hierarchyId":"bbed9f59-0412-4585-
a61d-6044db526aee","associationType":"CHILD"}
```

To view your data in the cold tier

1. Navigate to the Amazon S3 console.
2. In the navigation pane, choose **Buckets**, and then choose your Amazon S3 bucket.
3. Navigate to the folder that contains the raw data, asset metadata, or asset hierarchy metadata.
4. Select the files, and then from **Actions**, choose **Download**.

Storage data index files

AWS IoT SiteWise uses these files to optimize data query performance. They appear in your Amazon S3 bucket, but you don't need to use them.

File path

AWS IoT SiteWise stores data index files in the cold tier using the following template.

```
keyPrefix/index/series=timeseriesId/startYear=startYear/startMonth=startMonth/
startDay=startDay/index_timeseriesId_startTimestamp_quality
```

`index_7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-d34e-43e1-bc6f-1b490154b07a_1643846400_GOOD`

.

You can also import and export asset metadata. For more information see

When you enable AWS IoT SiteWise to export data to the cold tier for the first time, asset metadata is exported to the cold tier. After the initial configuration, AWS IoT SiteWise exports asset metadata to the tier only when you change asset model definitions or asset definitions. Asset metadata is saved in the cold tier in the newline delimited JSON (`.ndjson`) format.

File path

AWS IoT SiteWise stores asset metadata in the cold tier using the following template.

```
{keyPrefix}/asset_metadata/asset_{assetId}.ndjson
```

Every file path to asset metadata in the cold tier contains the following components.

Path component	Description
`keyPrefix`	The Amazon S3 prefix that you specified in the AWS IoT SiteWises storage configuration. Amazon S3 uses the prefix as a folder name in the bucket.
`asset_metadata`	The folder that stores asset metadata. The `asset_metadata` folder is saved in the prefix folder.
`fileName`	The file name uses the underscore (_) character as a delimiter to separate the following: • The `asset` prefix. • The `assetId` value.
	The file is saved in the `.ndjson` format.

Example file path to asset metadata in the colder tier

```
keyPrefix/asset_metadata/asset_35901915-d476-4dca-8637-d9ed4df939ed.ndjson
```

Fields

The schema of asset metadata that is exported to the cold tier contains the following fields.

`assetId`	The ID of the asset.
`assetName`	The name of the asset.
`assetExternalId`	The external ID of the asset.
`assetModelId`	The ID of the asset model used to create this asset.
`assetModelName`	The name of the asset model.
`assetModelExternalId`	The external ID of the asset model.
`assetPropertyId`	The ID of the asset property.
`assetPropertyName`	The name of the asset property.
`assetPropertyExternalId`	The external ID of the asset property.
`assetPropertyDataType`	The data type of the asset property.
`assetPropertyUnit`	The unit of the asset property (for example, `Newtons` and `RPM`).
`assetPropertyAlias`	The alias that identifies the asset property, such as an OPC-UA server data stream path (for example, `/company/windfarm/3/turbine/7/temperature`).
`timeSeriesId`	The ID that identifies the time series data from equipment, metrics, or transforms. You

	can use this field to join raw data and asset metadata in queries.
`timeSeriesBucket`	A hexadecimal number between 00 and ff. This number is derived from `timeSeriesId` .
	This partition is used to increase throughpu
	t when AWS IoT SiteWise writes to the cold
	tier. When you use Amazon Athena to run
	queries, you can use the partition for fine-grai n partitioning to improve query performance.
	`timeSeriesBucket` and `seriesBucket` in the file path to raw data are the same number.
`assetCompositeModelId`	The ID of the composite model.
`assetCompositeModelExternalId`	The external ID of the composite model.
`assetCompositeModelDescription`	The description of the composite model.
`assetCompositeModelName`	The name of the composite model.
`assetCompositeModelType`	The type of the composite model. For alarm composite models, this type is `AWS/ALARM` .
`assetCreationDate`	The date the asset was created, in Unix epoch time.
`assetLastUpdateDate`	The date the asset was last updated, in Unix epoch time.
`assetStatusErrorCode`	The error code.
`assetStatusErrorMessage`	The error message.
`assetStatusState`	The current status of the asset.
Field name	**Description**

Example asset metadata in the cold tier


```
 {"assetId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d","assetExternalId":null,"assetName":"Wind Turbine Asset
 2","assetModelId":"ec1d924f-f07d-444f-b072-
e2994c165d35","assetModelExternalId":null,"assetModelName":"Wind
 Turbine Asset Model","assetPropertyId":"95e63da7-d34e-43e1-
bc6f-1b490154b07a","assetPropertyExternalId":null,"assetPropertyName":"Temperature","assetPrope
Washington/Seattle/WT2/temp","timeSeriesId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeSeriesBucket":"f6","assetArn":null,"assetCompositeModelDescription":nul
 {"assetId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d","assetExternalId":null,"assetName":"Wind Turbine Asset
 2","assetModelId":"ec1d924f-f07d-444f-b072-
e2994c165d35","assetModelExternalId":null,"assetModelName":"Wind Turbine Asset
 Model","assetPropertyId":"c706d54d-4c11-42dc-9a01-63662fc697b4","assetPropertyExternalId":null
Washington/Seattle/WT2/pressure","timeSeriesId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d_c706d54d-4c11-42dc-9a01-63662fc697b4","timeSeriesBucket":"1e","assetArn":null,"ass
 {"assetId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d","assetExternalId":null,"assetName":"Wind Turbine Asset
 2","assetModelId":"ec1d924f-f07d-444f-b072-
e2994c165d35","assetModelExternalId":null,"assetModelName":"Wind
 Turbine Asset Model","assetPropertyId":"8cf1162f-dead-4fbe-b468-
```

Integrate with other services

AWS IoT SiteWise integrates with several AWS services to develop a complete AWS IoT solution in the AWS Cloud. For more information, see Interacting with other AWS services

AWS IoT SiteWise concepts

The following are the core concepts of AWS IoT SiteWise:

Aggregate

Aggregates are fundamental metrics, or measurements, that AWS IoT SiteWise automatically calculates for all time series data. For more information, see Querying asset property aggregates.

Asset

When you input, or ingest, data into AWS IoT SiteWise from your industrial equipment, your devices, equipment, and processes are each shown as assets. Each asset has associated data. For example, a piece of equipment might have a serial number, a location, a make and model, and an installation date. It might also have time series values for availability, performance, quality, temperature, pressure, and more. Group assets into hierarchies, allowing assets to access data stored in their child assets. For more information, see Modeling industrial assets.

Asset hierarchy

Set up asset hierarchies to create logical representations of your industrial operations. To do this, define a hierarchy in an asset model and associate assets created from that model with the specified hierarchy. Metrics in parent assets can combine data from the properties of child assets, allowing you to calculate metrics that offer insights into your overall operation or a specific part of it. For more information, see Defining asset model hierarchies.

Asset model

Every asset is made using an asset model. Asset models are structures that define and standardize the format of your assets. They ensure consistent information across multiple assets of the same type, allowing you to handle data in assets that represent groups of devices. In each asset model, you can define attributes, time series inputs (measurements), time series transformations (transforms), time series aggregations (metrics), and asset hierarchies. For more information, see Modeling industrial assets.

Decide where your asset model's properties are processed by configuring your asset model for the edge. Utilize this feature to handle and monitor asset data on your local devices.

Asset property

Asset properties are the structures within each asset that hold industrial data. Each property has a data type and can also have a unit. A property can be an attribute, a measurement, a transform, or a metric. For more information, see Defining data properties.

Configure asset properties to compute at the edge. For more information about processing data at the edge, see the section called “Enabling edge data processing”.

Attribute

Attributes are properties of an asset that typically stay constant, like the device manufacturer or device location. Attributes can have preset values. Every asset created from an asset model includes the default values of the attributes defined in that model. For more information, see Defining static data (attributes).

Dashboard

Each project contains a set of dashboards. Dashboards provide a set of visualizations for the values of a set of assets. Project owners create the dashboards and the visualizations that it contains. When a project owner is ready to share the set of dashboards, the owner can invite viewers to the project, which gives them access to all dashboards in the project. If you want a different set of viewers for different dashboards, you must divide the dashboards between projects. When viewers look at dashboards, they can customize time range to look at specific data.

Data stream

Input, or ingest, industrial data into AWS IoT SiteWise even before creating asset models and assets. AWS IoT SiteWise automatically generates data streams to collect raw data streams from your equipment.

Data stream alias

Data stream aliases help you easily identify a data stream. For example, the alias `server1-windfarm/3/turbine/7/temperature` indicates temperature values coming from turbine #7 in wind farm #3. The term `server1` is the data source name that helps identify the OPC-UA server, and `server1-` is a prefix attached to all data streams reported from this OPC-UA server.

Data stream association

After you create asset models and assets, associate data streams with asset properties defined in your assets to structure your data. AWS IoT SiteWise can then use asset models and assets to handle incoming data from your data streams. You can also disassociate data streams from asset properties. For more information, see Managing data streams.

Formula

Each transform and metric property comes with a formula that outlines how the property transforms or aggregates data. These formulas include property inputs, operators, and functions offered by AWS IoT SiteWise. For more information, see Using formula expressions.

Measurement

Measurements are properties of an asset that depict the raw sensor time series data streams from a device or equipment. For more information, see Defining data streams from equipment (measurements).

Metric

Metrics are properties of an asset that represent aggregated time series data. Each metric is accompanied by a mathematical expression (formula) that outlines how to aggregate data points and a time interval for computing that aggregation. Metrics generate a single data point for each specified time interval. For more information, see Aggregating data from properties and other assets (metrics).

Packs

SiteWise Edge gateways use packs to determine how to collect, process, and route data. Currently, AWS IoT SiteWise supports the data collection pack and the data processing pack. For more information about the available packs for your SiteWise Edge gateway, see the section called “Using packs”.

Data collection pack

Use the data collection pack so that your SiteWise Edge gateway can collect your industrial data and route it to the AWS destination of your choice. This pack is automatically added to your SiteWise Edge gateway and can't be removed.

Data processing pack

Use the data processing pack to process your data at the edge and retain it for 30 days for use in local applications.

Portal

An AWS IoT SiteWise Monitor portal is a web application that you can use to visualize and share your AWS IoT SiteWise data. A portal has one or more administrators and contains zero or more projects.

Portal administrator

Each SiteWise Monitor portal has one or more portal administrators. Portal administrators use the portal to create projects that contain collections of assets and dashboards. The portal administrator then assigns assets and owners to each project. By controlling access to the project, portal administrators specify which assets that project owners and viewers can see.

Project

Each SiteWise Monitor portal contains a set of projects. Each project has a subset of your AWS IoT SiteWise assets associated with it. Project owners create one or more dashboards to provide a consistent way to view the data associated with those assets. Project owners can invite viewers to the project to allow them to view the assets and dashboards in the project. The project is the basic unit of sharing within SiteWise Monitor. Project owners can invite users who were given access to the portal by the AWS administrator. A user must have access to a portal before a project in that portal can be shared with that user.

Project owner

Each SiteWise Monitor project has owners. Project owners create visualizations in the form of dashboards to represent operational data in a consistent manner. When dashboards are ready to share, the project owner can invite viewers to the project. Project owners can also assign other owners to the project. Project owners can configure thresholds and notification settings for alarms.

Project viewer

Each SiteWise Monitor project has viewers. Project viewers can connect to the portal to view the dashboards that project owners created. In each dashboard, project viewers can adjust the time range to better understand operational data. Project viewers can only view dashboards in the projects to which they have access. Project viewers can acknowledge and snooze alarms.

Property alias

You have the option to create aliases on asset properties, such as an OPC-UA server data stream path (for example, /company/windfarm/3/turbine/7/temperature), simplifying the

identification of an asset property during the ingestion or retrieval of asset data. When you use a SiteWise Edge gateway to ingest data from servers, your property aliases must match the paths of your raw data streams. For more information, see Mapping industrial data streams to asset properties.

Property notification

When you enable property notifications for an asset property, AWS IoT SiteWise publishes an MQTT message to AWS IoT Core each time that property receives a new value. The message payload includes details about the update to that property value. Use property value notifications to create solutions that connect your industrial data in AWS IoT SiteWise with other AWS services. For more information, see Interacting with other AWS services.

SiteWise Edge gateway

A SiteWise Edge gateway is situated on the customer's premises to gather, handle, and direct data. A SiteWise Edge gateway connects to your industrial data sources through OPC-UA protocol to gather and process data, sending it to the AWS cloud. SiteWise Edge gateways can also connect to partner data sources. SiteWise Edge gateways use packs for data collection, edge processing, and more. For more information about available packs, see the section called "Using packs".

You have the flexibility to create a SiteWise Edge gateway on any device or platform capable of running AWS IoT Greengrass. For more information, see Using SiteWise Edge gateways.

Transform

Transforms are properties of an asset that represent transformed time series data. Every transform is accompanied by a mathematical expression (formula) that specifies how to convert data points from one form to another. The transformed data points hold a one-to-one relationship with the input data points. For more information, see Transforming data (transforms).

Visualization

In each dashboard, project owners decide how to display the properties and alarms of the assets associated with the project. Availability might be represented as a line chart, while other values might be displayed as bar charts or key performance indicators (KPIs). Alarms are best displayed as status grids and status timelines. Project owners customize each visualization to provide the best understanding of the data for that asset.

Use cases for AWS IoT SiteWise

AWS IoT SiteWise is used across a variety of industries for many industrial data collection and analysis applications.

Collect data consistently from all your sources to help resolve issues quickly. AWS IoT SiteWise offers remote monitoring to collect the data directly on-site or gather it from multiple sources across many facilities. AWS IoT SiteWise provides the necessary flexibility for industrial IoT data solutions.

Manufacturing

AWS IoT SiteWise can simplify the process of collecting and utilizing data from your equipment to pinpoint and minimize inefficiencies, enhancing industrial operations. AWS IoT SiteWise helps you collect data from manufacturing lines and equipment. With AWS IoT SiteWise, you can transfer the data to the AWS Cloud and build performance metrics for your specific equipment and processes. You can use the metrics produced to understand the overall effectiveness of your operations and identify opportunities for innovation and improvement. You can also view your manufacturing process and identify equipment and process deficiencies, production gaps, or product defects.

Food and beverage

Food and beverage industry facilities handle a wide variety of food processing, including grinding grain to flour, butchering and packing meat, and assembling, cooking, and freezing microwaveable meals. Food processing plants often span multiple locations with plant and equipment operators in a centralized location to monitor processes and equipment. For example, refrigeration units assess ingredient handling and expiration. They monitor waste creation across facilities to ensure operational efficiency. With AWS IoT SiteWise, you can group sensor data streams from multiple locations by production line, and facilities so your process engineers can better understand and make improvements across facilities.

Energy and utilities

With AWS IoT SiteWise, you can resolve equipment issues easier and more efficiently. You can monitor asset performance remotely and in real time. Access historical equipment data from anywhere to pinpoint potential problems, dispatch accurate resources, and both prevent and fix issues faster.

Getting started with AWS IoT SiteWise

With AWS IoT SiteWise, you can collect, organize, analyze, and visualize your data.

AWS IoT SiteWise provides a demo that you can use to explore the service without configuring a real data source. For more information, see Using the AWS IoT SiteWise demo.

You can complete the following tutorials to explore certain features of AWS IoT SiteWise:

- Ingesting data from AWS IoT things
- Visualizing and sharing wind farm data in SiteWise Monitor
- Publishing property value updates to Amazon DynamoDB

See the following topics to learn more about AWS IoT SiteWise:

- Ingesting data to AWS IoT SiteWise
- Modeling industrial assets
- Enabling edge data processing
- Monitoring data with AWS IoT SiteWise Monitor
- Query data from AWS IoT SiteWise
- Interacting with other AWS services

Topics

Requirements

You must have an AWS account to get started with AWS IoT SiteWise. If you don't have one, see Setting up an AWS account.

Use a Region where AWS IoT SiteWise is available. For more information, see AWS IoT SiteWise endpoints and quotas. You can use the Region selector in the AWS Management Console to switch to one of these Regions.

Setting up an AWS account

Topics

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to an administrative user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create an administrative user

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create an administrative user

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to an administrative user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the administrative user

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Using the AWS IoT SiteWise demo

You can easily explore AWS IoT SiteWise by using the AWS IoT SiteWise demo. AWS IoT SiteWise provides the demo as an AWS CloudFormation template that you can deploy to create asset models, assets, and a SiteWise Monitor portal, and generate sample data for up to a week.

Important

Once you create the demo, you will start being charged for the resources that this demo creates and consumes.

Topics

Creating the AWS IoT SiteWise demo

You can create the AWS IoT SiteWise demo from the AWS IoT SiteWise console.

Note

The demo creates Lambda functions, one CloudWatch Events rule, and the AWS Identity and Access Management (IAM) roles required for the demo. You might see these resources in your AWS account. We recommend that you keep these resources until you're done with the demo. If you delete the resources, the demo might stop working correctly.

To create the demo in the AWS IoT SiteWise console

1. Navigate to the AWS IoT SiteWise console and find the **SiteWise demo** in the upper-right corner of the page.
2. (Optional) Under **SiteWise demo**, change the **Days to keep demo assets** field to specify how many days to keep the demo before deleting it.
3. (Optional) To create a SiteWise Monitor portal to monitor sample data, do the following.

   Note

   You will be charged for the SiteWise Monitor resources that this demo creates and consumes. For more information, see SiteWise Monitor in the *AWS IoT SiteWise Pricing*.

   a. Choose **Monitor Resources**.

b. Choose **Permission**.

c. Choose an existing IAM role that grants your federated IAM users access to the portal.

> **Important**
>
> Your IAM role must have the following permissions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:Describe*",
                "iotsitewise:List*",
                "iotsitewise:Get*",
                "cloudformation:DescribeStacks",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedRolePolicies",
                "sso:DescribeRegisteredRegions",
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        }
    ]
}
```

For more information about how to work with SiteWise Monitor, see What is AWS IoT SiteWise Monitor? in the *AWS IoT SiteWise Monitor Application Guide*.

4. Choose **Create demo**.

The demo takes around 3 minutes to create. If the demo fails to create, your account might have insufficient permissions. Switch to an account that has administrative permissions, or use the following steps to delete the demo and try again:

a. Choose **Delete demo**.

The demo takes around 15 minutes to delete.

   b. If the demo doesn't delete, open the AWS CloudFormation console, choose the stack named **IoTSiteWiseDemoAssets**, and choose **Delete** in the upper-right corner.
   c. If the demo fails to delete again, follow the steps in the AWS CloudFormation console to skip the resources that failed to delete, and try again.
5. After the demo creates successfully, you can explore the demo assets and data in the AWS IoT SiteWise console.

Deleting the AWS IoT SiteWise demo

The AWS IoT SiteWise demo deletes itself after a week, or the number of days you chose if you created the demo stack from the AWS CloudFormation console. You can delete the demo before if you're done using the demo resources. You can also delete the demo if the demo fails to create. Use the following steps to delete the demo manually.

To delete the AWS IoT SiteWise demo

1. Navigate to the AWS CloudFormation console.
2. Choose **IoTSiteWiseDemoAssets** from the list of **Stacks**.
3. Choose **Delete**.

   When you delete the stack, all of the resources created for the demo are deleted.
4. In the confirmation dialog, choose **Delete stack**.

   The stack takes around 15 minutes to delete. If the demo fails to delete, choose **Delete** in the upper-right corner again. If the demo fails to delete again, follow the steps in the AWS CloudFormation console to skip the resources that failed to delete, and try again.

AWS IoT SiteWise tutorials

Welcome to the AWS IoT SiteWise tutorials page. This growing collection of tutorials empowers you with the knowledge and skills needed to navigate the intricacies of AWS IoT SiteWise. These tutorials offer a diverse range of basic topics to cater to your needs. As you delve into the tutorials, uncover invaluable insights into various aspects of AWS IoT SiteWise.

Each tutorial uses a specific equipment example. These tutorials are intended for test environments, and they use fictitious company names, models, assets, properties, and so on. Their purpose is to provide general guidance. The tutorials are not intended for direct use in a production environment without careful review and adaptation to meet the unique needs of your organization.

Topics

Calculating OEE in AWS IoT SiteWise

This tutorial provides an example of how to calculate overall equipment effectiveness (OEE) for a manufacturing process. As a result, your OEE calculations or formulas might differ from those shown here. In general, OEE is defined as `Availability * Quality * Performance`. To learn more about calculating OEE, see Overall equipment effectiveness on *Wikipedia*.

Prerequisites

To complete this tutorial, you must configure data ingestion for a device that has the following three data streams:

- `Equipment_State` – A numerical code that represents the state of the machine, such as idle, fault, planned stop, or normal operation.
- `Good_Count` – A data stream where each data point contains the number of successful operations since the last data point.
- `Bad_Count` – A data stream where each data point contains the number of unsuccessful operations since the last data point.

To configure data ingestion, see Ingesting data to AWS IoT SiteWise. If you don't have an available industrial operation, you can write a script that generates and uploads sample data through the AWS IoT SiteWise API.

How to calculate OEE

In this tutorial, you create an asset model that calculates OEE from three data input streams: `Equipment_State`, `Good_Count`, and `Bad_Count`. In this example, consider a generic packaging machine, such as one that's used for packaging sugar, potato chips, or paint. In the AWS IoT SiteWise console, create an AWS IoT SiteWise asset model with the following measurements, transforms, and metrics. Then, you can create an asset to represent the packaging machine and observe how AWS IoT SiteWise calculates OEE.

Define the following measurements to represent the raw data streams from the packaging machine.

Measurements

- `Equipment_State` – A data stream (or measurement) that provides the current state of the packaging machine in numerical codes:
  - `1024` – The machine is idle.
  - `1020` – A fault, such as an error or delay.
  - `1000` – A planned stop.
  - `1111` – A normal operation.
- `Good_Count` – A data stream where each data point contains the number of successful operations since the last data point.
- `Bad_Count` – A data stream where each data point contains the number of unsuccessful operations since the last data point.

Using the `Equipment_State` measurement data stream and the codes it contains, define the following transforms (or derived measurements). Transforms have a one-to-one relationship with raw measurements.

Transforms

- `Idle = eq(Equipment_State, 1024)` – A transformed data stream that contains the machine's idle state.

- `Fault = eq(Equipment_State, 1020)` – A transformed data stream that contains the machine's fault state.
- `Stop = eq(Equipment_State, 1000)` – A transformed data stream that contains the machine's planned stop state.
- `Running = eq(Equipment_State, 1111)` – A transformed data stream that contains the machine's normal operational state.

Using the raw measurements and the transformed measurements, define the following metrics that aggregate machine data over specified time intervals. Choose the same time interval for each metric when you define the metrics in this section.

Metrics

- `Successes = sum(Good_Count)` – The number of successfully filled packages over the specified time interval.
- `Failures = sum(Bad_Count)` – The number of unsuccessfully filled packages over the specified time interval.
- `Idle_Time = statetime(Idle)` – The machine's total idle time (in seconds) per specified time interval.
- `Fault_Time = statetime(Fault)` – The machine's total fault time (in seconds) per specified time interval.
- `Stop_Time = statetime(Stop)` – The machine's total planned stop time (in seconds) per specified time interval.
- `Run_Time = statetime(Running)` – The machine's total time (in seconds) running without issue per specified time interval.
- `Down_Time = Idle_Time + Fault_Time + Stop_Time` – The machine's total downtime (in seconds) over the specified time interval, calculated as the sum of the machine states other than `Run_Time`.
- `Availability = Run_Time / (Run_Time + Down_Time)` – The machine's uptime or percentage of scheduled time that the machine is available to operate over the specified time interval.
- `Quality = Successes / (Successes + Failures)` – The machine's percentage of successfully filled packages over the specified time intervals.

- `Performance = ((Successes + Failures) / Run_Time) / Ideal_Run_Rate` – The machine's performance over the specified time interval as a percentage out of the ideal run rate (in seconds) for your process.

  For example, your `Ideal_Run_Rate` might be 60 packages per minute (1 package per second). If your `Ideal_Run_Rate` is per minute or per hour, you need to divide it by the appropriate unit conversion factor because `Run_Time` is in seconds.
- `OEE = Availability * Quality * Performance` – The machine's overall equipment effectiveness over the specified time interval. This formula calculates OEE as a fraction out of 1.

Ingesting data from AWS IoT things

Learn how to ingest data to AWS IoT SiteWise from a fleet of AWS IoT things by using device shadows in this tutorial. *Device shadows* are JSON objects that store current state information for an AWS IoT device. For more information, see Device shadow service in the *AWS IoT Developer Guide*.

After you complete this tutorial, you can set up an operation in AWS IoT SiteWise based on AWS IoT things. By using AWS IoT things, you can integrate your operation with other useful features of AWS IoT. For example, you can configure AWS IoT features to do the following tasks:

- Configure additional rules to stream data to AWS IoT Events, Amazon DynamoDB, and other AWS services. For more information, see Rules in the *AWS IoT Developer Guide*.
- Index, search, and aggregate your device data with the AWS IoT fleet indexing service. For more information, see Fleet indexing service in the *AWS IoT Developer Guide*.
- Audit and secure your devices with AWS IoT Device Defender. For more information, see AWS IoT Device Defender in the *AWS IoT Developer Guide*.

In this tutorial, you learn how to ingest data from AWS IoT things' device shadows to assets in AWS IoT SiteWise. To do so, you create one or more AWS IoT things and run a script that updates each thing's device shadow with CPU and memory usage data. You use CPU and memory usage data in this tutorial to imitate realistic sensor data. Then, you create a rule with an AWS IoT SiteWise action that sends this data to an asset in AWS IoT SiteWise every time a thing's device shadow updates. For more information, see Ingesting data using AWS IoT Core rules.

Topics

Prerequisites

To complete this tutorial, you need the following:

- An AWS account. If you don't have one, see Setting up an AWS account.
- A development computer running Windows, macOS, Linux, or Unix to access the AWS Management Console. For more information, see Getting Started with the AWS Management Console.
- An AWS Identity and Access Management (IAM) user with administrator permissions.
- Python 3 installed on your development computer or installed on the device that you want to register as an AWS IoT thing.

Step 1: Creating an AWS IoT policy

In this procedure, create an AWS IoT policy that allows your AWS IoT things to access the resources used in this tutorial.

To create an AWS IoT policy

1. Sign in to the AWS Management Console.
2. Review the AWS Regions where AWS IoT SiteWise is supported. Switch to one of these supported Regions, if necessary.
3. Navigate to the AWS IoT console. If a **Connect device** button appears, choose it.

4. In the left navigation pane, choose **Security** and then choose **Policies**.
5. Choose **Create**.
6. Enter a name for the AWS IoT policy (for example, **`SiteWiseTutorialDevicePolicy`**).
7. Under **Policy document**, choose **JSON** to enter the following policy in JSON form. Replace *`region`* and *`account-id`* with your Region and account ID, such as **`us-east-1`** and **`123456789012`**.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:region:account-id:client/SiteWiseTutorialDevice*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": [
        "arn:aws:iot:region:account-id:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/update",
        "arn:aws:iot:region:account-id:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/delete",
        "arn:aws:iot:region:account-id:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/get"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": [
        "arn:aws:iot:region:account-id:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/update/accepted",
        "arn:aws:iot:region:account-id:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/delete/accepted",
        "arn:aws:iot:region:account-id:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/get/accepted",
        "arn:aws:iot:region:account-id:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/update/rejected",
        "arn:aws:iot:region:account-id:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/delete/rejected"
      ]
```

```
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/update/accepted",
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/delete/accepted",
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/get/accepted",
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/update/rejected",
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/delete/rejected"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:DeleteThingShadow"
      ],
      "Resource": "arn:aws:iot:region:account-id:thing/SiteWiseTutorialDevice*"
    }
  ]
}
```

This policy enables your AWS IoT devices to establish connections and communicate with device shadows using MQTT messages. For more information about MQTT messages, see What is MQTT?. To interact with device shadows, your AWS IoT things publish and receive MQTT messages on topics that start with `$aws/things/`*`thing-name`*`/shadow/`. This policy incorporates a thing policy variable known as `${iot:Connection.Thing.ThingName}`. This variable substitutes the connected thing's name in each topic. The `iot:Connect` statement sets limitations on which devices can establish connections, ensuring that the thing policy variable can only substitute names starting with `SiteWiseTutorialDevice`.

For more information, see Thing policy variables in the *AWS IoT Developer Guide*.

> **Note**
>
> This policy applies to things whose names start with `SiteWiseTutorialDevice`. To use a different name for your things, you must update the policy accordingly.

8. Choose **Create**.

Step 2: Creating and configuring an AWS IoT thing

In this procedure, you create and configure an AWS IoT thing. You can designate your development computer as an AWS IoT thing. As you progress, remember that the principles you're learning here can be applied to actual projects. You have the flexibility to make and set up AWS IoT things on any device capable of running an AWS IoT SDK, including AWS IoT Greengrass and FreeRTOS. For more information, see AWS IoT SDKs in the *AWS IoT Developer Guide*.

To create and configure an AWS IoT thing

1. Open a command line and run the following command to create a directory for this tutorial.

```
mkdir iot-sitewise-rule-tutorial
cd iot-sitewise-rule-tutorial
```

2. Run the following command to create a directory for your thing's certificates.

```
mkdir device1
```

   If you're creating additional things, increment the number in the directory name accordingly to keep track of which certificates belong to which thing.

3. Navigate to the AWS IoT console.
4. In the left navigation pane, choose **All devices** in the **Manage** section. Then choose **Things**.
5. If a **You don't have any things yet** dialog box appears, choose **Create a thing**. Otherwise, choose **Create things**.
6. On the **Creating things** page, choose **Create a single thing** and then choose **Next**.
7. On the **Specify thing properties** page, enter a name for your AWS IoT thing (for example, **`SiteWiseTutorialDevice1`**) and then choose **Next**. If you're creating additional things, increment the number in the thing name accordingly.

> **Important**
>
> The thing name must match the name used in the policy that you created in *Step 1: Creating an AWS IoT policy*. Otherwise, your device can't connect to AWS IoT.

8. On the **Configure device certificate - *optional*** page, choose **Auto-generate a new certificate (recommended)** the choose **Next**. Certificates enable AWS IoT to securely identify your devices.
9. On the **Attach policies to certificate - *optional*** page, select the policy you created in *Step 1: Creating an AWS IoT policy* and choose **Create thing**.
10. On the **Download certificates and keys** dialog box, do the following:

    a. Choose the **Download** links to download your thing's certificate, public key, and private key. Save all three files to the directory that you created for your thing's certificates (for example, `iot-sitewise-rule-tutorial/device1`).

    > **Important**
    >
    > This is the only time that you can download your thing's certificate and keys, which you need for your device to successfully connect to AWS IoT.

    b. Choose the **Download** link to download a root CA certificate. Save the root CA certificate to the `iot-sitewise-rule-tutorial`. We recommend downloading Amazon Root CA 1.
11. Choose **Done**.

You have now registered an AWS IoT thing on your computer. Take one of the following next steps:

- Continue to *Step 3: Creating a device asset model* without creating additional AWS IoT things. You can complete this tutorial with only one thing.
- Repeat the steps in this section on another computer or device to create more AWS IoT things. For this tutorial, we recommend that you follow this option so that you can ingest unique CPU and memory usage data from multiple devices.
- Repeat the steps in this section on the same device (your computer) to create more AWS IoT things. Each AWS IoT thing receives similar CPU and memory usage data from your computer, so use this approach to demonstrate ingesting non-unique data from multiple devices.

Step 3: Creating a device asset model

In this procedure, you create an asset model in AWS IoT SiteWise to represent your devices that stream CPU and memory usage data. To process data in assets that represent groups of devices, asset models enforce consistent information across multiple assets of the same type. For more information, see Modeling industrial assets.

To create an asset model that represents a device

1. Navigate to the AWS IoT SiteWise console.
2. In the left navigation pane, choose **Models**.
3. Choose **Create model**.
4. Under **Model details**, enter a name for your model. For example, **`SiteWise Tutorial Device Model`**.
5. Under **Measurement definitions**, do the following:

   a. In **Name**, enter **`CPU Usage`**.

   b. In **Unit**, enter **`%`**.

   c. Leave the **Data type** as **Double**.

   Measurement properties represent a device's raw data streams. For more information, see Defining data streams from equipment (measurements).
6. Choose **Add new measurement** to add a second measurement property.
7. In the second row under **Measurement definitions**, do the following:

   a. In **Name**, enter **`Memory Usage`**.

   b. In **Unit**, enter **`%`**.

   c. Leave the **Data type** as **Double**.
8. Under **Metric definitions**, do the following:

   a. In **Name**, enter **`Average CPU Usage`**.

   b. In **Formula**, enter **`avg(CPU Usage)`**. Choose **CPU Usage** from the autocomplete list when it appears.

   c. In **Time interval**, enter **`5 minutes`**.

Metric properties define aggregation calculations that process all input data points over an interval and output a single data point per interval. This metric property calculates each device's average CPU usage every 5 minutes. For more information, see Aggregating data from properties and other assets (metrics).

9. Choose **Add new metric** to add a second metric property.
10. In the second row under **Metric definitions**, do the following:

    a. In **Name**, enter **`Average Memory Usage`**.

    b. In **Formula**, enter **`avg(Memory Usage)`**. Choose **Memory Usage** from the autocomplete list when it appears.

    c. In **Time interval**, enter **`5 minutes`**.

    This metric property calculates each device's average memory usage every 5 minutes.
11. (Optional) Add other additional metrics that you're interested in calculating per device. Some interesting functions include `min` and `max`. For more information, see Using formula expressions. In *Step 4: Creating a device fleet asset model*, you create a parent asset that can calculate metrics using data from your entire fleet of devices.
12. Choose **Create model**.

Step 4: Creating a device fleet asset model

In this procedure, you craft an asset model in AWS IoT SiteWise to symbolize your collection of devices. Within this asset model, you establish a structure that allows you to link numerous device assets to one overarching fleet asset. Following that, you outline metrics in the fleet asset model to consolidate data from all connected device assets. This approach provides you with comprehensive insights into the collective performance of your entire fleet.

To create an asset model that represents a device fleet

1. Navigate to the AWS IoT SiteWise console.
2. In the left navigation pane, choose **Models**.
3. Choose **Create model**.
4. Under **Model details**, enter a name for your model. For example, **`SiteWise Tutorial Device Fleet Model`**.

5. Under **Hierarchy definitions**, do the following:

   a. In **Hierarchy name**, enter **`Device`**.

   b. In **Hierarchy model**, choose your device asset model (**`SiteWise Tutorial Device Model`**).

   A hierarchy defines a relationship between a parent (fleet) asset model and a child (device) asset model. Parent assets can access child assets' property data. When you create assets later, you need to associate child assets to parent assets according to a hierarchy definition in the parent asset model. For more information, see Defining asset model hierarchies.

6. Under **Metric definitions**, do the following:

   a. In **Name**, enter **`Average CPU Usage`**.

   b. In **Formula**, enter **`avg(Device | Average CPU Usage)`**. When the autocomplete list appears, choose **Device** to choose a hierarchy, then choose **Average CPU Usage** to choose the metric from the device asset that you created earlier.

   c. In **Time interval**, enter **`5 minutes`**.

   This metric property calculates the average CPU usage of all device assets associated to a fleet asset through the **`Device`** hierarchy.

7. Choose **Add new metric** to add a second metric property.

8. In the second row under **Metric definitions**, do the following:

   a. In **Name**, enter **`Average Memory Usage`**.

   b. In **Formula**, enter **`avg(Device | Average Memory Usage)`**. When the autocomplete list appears, choose **Device** to choose a hierarchy, then choose **Average Memory Usage** to choose the metric from the device asset that you created earlier.

   c. In **Time interval**, enter **`5 minutes`**.

   This metric property calculates the average memory usage of all device assets associated to a fleet asset through the **`Device`** hierarchy.

9. (Optional) Add other additional metrics that you're interested in calculating across your fleet of devices.

10. Choose **Create model**.

Step 5: Creating and configuring a device asset

In this procedure, you generate a device asset that's based on your device asset model. Then, you define property aliases for each measurement property. A *property alias* is a unique string that identifies an asset property. Later, you can identify a property for data upload by using the aliases instead of the asset ID and property ID. For more information, see Mapping industrial data streams to asset properties.

To create a device asset and define property aliases

1. Navigate to the AWS IoT SiteWise console.
2. In the left navigation pane, choose **Assets**.
3. Choose **Create asset**.
4. Under **Model information**, choose your device asset model, **`SiteWise Tutorial Device Model`**.
5. Under **Asset information**, enter a name for your asset. For example, **`SiteWise Tutorial Device 1`**.
6. Choose **Create asset**.
7. For your new device asset, choose **Edit**.
8. Under **CPU Usage**, enter **`/tutorial/device/SiteWiseTutorialDevice1/cpu`** as the property alias. You include the AWS IoT thing's name in the property alias, so that you can ingest data from all of your devices using a single AWS IoT rule.
9. Under **Memory Usage**, enter **`/tutorial/device/SiteWiseTutorialDevice1/memory`** as the property alias.
10. Choose **Save**.

If you created multiple AWS IoT things earlier, repeat steps 3 through 10 for each device, and increment the number in the asset name and property aliases accordingly. For example, the second device asset's name should be **`SiteWise Tutorial Device 2`**, and its property aliases should be **`/tutorial/device/SiteWiseTutorialDevice2/cpu`**, and **`/tutorial/device/SiteWiseTutorialDevice2/memory`**.

Step 6: Creating and configuring a device fleet asset

In this procedure, you form a device fleet asset derived from your device fleet asset model. Then, you link your individual device assets to the fleet asset. This association enables the metric

properties of the fleet asset to compile and analyze data from multiple devices. This data provides you with a consolidated view of the collective performance of the entire fleet.

To create a device fleet asset and associate device assets

1. Navigate to the AWS IoT SiteWise console.
2. In the left navigation pane, choose **Assets**.
3. Choose **Create asset**.
4. Under **Model information**, choose your device fleet asset model, **`SiteWise Tutorial Device Fleet Model`**.
5. Under **Asset information**, enter a name for your asset. For example, **`SiteWise Tutorial Device Fleet 1`**.
6. Choose **Create asset**.
7. For your new device fleet asset, choose **Edit**.
8. Under **Assets associated to this asset**, choose **Add associated asset** and do the following:
   a. Under **Hierarchy**, choose **Device**. This hierarchy identifies the hierarchical relationship between device and device fleet assets. You defined this hierarchy in the device fleet asset model earlier in this tutorial.
   b. Under **Asset**, choose your device asset, **SiteWise Tutorial Device 1**.
9. (Optional) If you created multiple device assets earlier, repeat steps 8 through 10 for each device asset that you created.
10. Choose **Save**.

    You should now see your device assets organized as a hierarchy.

Step 7: Creating a rule in AWS IoT Core to send data to device assets

In this procedure, you establish a rule in AWS IoT Core. The rule is designed to interpret notification messages from device shadows and transmit the data to your device assets in AWS IoT SiteWise.Each time your device's shadow updates, AWS IoT sends an MQTT message. You can create a rule that takes action when device shadows change based on the MQTT message. In this case, the aim is to handle the update message, extract the property values, and transmit them to your device assets in AWS IoT SiteWise.

To create a rule with an AWS IoT SiteWise action

1. Navigate to the AWS IoT console.
2. In the left navigation pane, choose **Message routing** and then choose **Rules**.
3. Choose **Create rule**.
4. Enter a name and description for your rule and the choose **Next**.
5. Enter the following SQL statement and the choose **Next**.

```
SELECT
  *
FROM
  '$aws/things/+/shadow/update/accepted'
WHERE
  startsWith(topic(3), "SiteWiseTutorialDevice")
```

   This rule query statement works because the device shadow service publishes shadow updates to `$aws/things/thingName/shadow/update/accepted`. For more information about device shadows, see Device shadow service in the *AWS IoT Developer Guide*.

   In the `WHERE` clause, this rule query statement uses the `topic(3)` function to get the thing name from the third segment of the topic. Then, the statement filters out devices that have names that don't match those of the tutorial devices. For more information about AWS IoT SQL, see AWS IoT SQL reference in the *AWS IoT Developer Guide*.
6. Under **Rule actions**, choose **Send message data to asset properties in AWS IoT SiteWise** and do the following:

   a. Choose **By property alias**.

   b. In **Property alias**, enter **`/tutorial/device/${topic(3)}/cpu`**.

      The `${...}` syntax is a substitution template. AWS IoT evaluates the contents within the braces. This substitution template pulls the thing name from the topic to create an alias unique to each thing. For more information, see Substitution templates in the *AWS IoT Developer Guide*.

      Note

      Because an expression in a substitution template is evaluated separately from the `SELECT` statement, you can't use a substitution template to reference an alias

created using an AS clause. You can reference only information present in the original payload, in addition to supported functions and operators.

c. In **Entry ID - *optional***, enter **`${concat(topic(3), "-cpu-", floor(state.reported.timestamp))}`**.

   Entry IDs uniquely identify each value entry attempt. If an entry returns an error, you can find the entry ID in the error output to troubleshoot the issue. The substitution template in this entry ID combines the thing name and the device's reported timestamp. For example, the resulting entry ID might look like `SiteWiseTutorialDevice1-cpu-1579808494`.

d. In **Time in seconds**, enter **`${floor(state.reported.timestamp)}`**.

   This substitution template calculates the time in seconds from the device's reported timestamp. In this tutorial, devices report timestamp in seconds in Unix epoch time as a floating point number.

e. In **Offset in nanos - *optional***, enter **`${floor((state.reported.timestamp % 1) * 1E9)}`**.

   This substitution template calculates the nanosecond offset from the time in seconds by converting the decimal portion of the device's reported timestamp.

   Note

   AWS IoT SiteWise requires that your data has a current timestamp in Unix epoch time. If your devices don't report time accurately, you can get the current time from the AWS IoT rules engine with timestamp(). This function reports time in milliseconds, so you must update your rule action's time parameters to the following values:

   - In **Time in seconds**, enter **`${floor(timestamp() / 1E3)}`**.
   - In **Offset in nanos**, enter **`${(timestamp() % 1E3) * 1E6}`**.

f. In **Data type**, choose **Double**.

   This data type must match the data type of the asset property you defined in the asset model.

g. In **Value**, enter **`${state.reported.cpu}`**. In substitution templates, you use the **`.`** operator to retrieve a value from within a JSON structure.

h. Choose **Add entry** to add a new entry for the memory usage property, and complete the following steps again for that property:

   i. Choose **By property alias**.

   ii. In **Property alias**, enter **`/tutorial/device/${topic(3)}/memory`**.

   iii. In **Entry ID - *optional***, enter **`${concat(topic(3), "-memory-", floor(state.reported.timestamp))}`**.

   iv. In **Time in seconds**, enter **`${floor(state.reported.timestamp)}`**.

   v. In **Offset in nanos - *optional***, enter **`${floor((state.reported.timestamp % 1) * 1E9)}`**.

   vi. In **Data type**, choose **Double**.

   vii. In **Value**, enter **`${state.reported.memory}`**.

i. Under **IAM Role**, choose **Create new role** to create an IAM role for this rule action. This role allows AWS IoT to push data to properties in your device fleet asset and its asset hierarchy.

j. Enter a role name and choose **Create**.

7. (Optional) Configure an error action that you can use to troubleshoot your rule. For more information, see Troubleshooting a rule.

8. Choose **Next**.

9. Review the settings and choose **Create** to create the rule.

Step 8: Running the device client script

For this tutorial, you aren't using an actual device to report data. Instead, you run a script to update your AWS IoT thing's device shadow with CPU and memory usage to imitate real sensor data. To run the script, you must first install required Python packages. In this procedure, you install the required Python packages and then run the device client script.

To configure and run the device client script

1. Navigate to the AWS IoT console.

2. At the bottom of the left navigation pane, choose **Settings**.

3. Save your custom endpoint for use with the device client script. You use this endpoint to interact with your thing's shadows. This endpoint is unique to your account in the current Region.

   Your custom endpoint should look like the following example.

   ```
   identifier.iot.region.amazonaws.com
   ```

4. Open a command line and run the following command to navigate to the tutorial directory you created earlier.

   ```
   cd iot-sitewise-rule-tutorial
   ```

5. Run the following command to install the AWS IoT Device SDK for Python.

   ```
   pip3 install AWSIoTPythonSDK
   ```

   For more information, see AWS IoT Device SDK for Python in the *AWS IoT Developer Guide*

6. Run the following command to install psutil, a cross-platform process and system utilities library.

   ```
   pip3 install psutil
   ```

   For more information, see psutil in the *Python Package Index*.

7. Create a file called `thing_performance.py` in the `iot-sitewise-rule-tutorial` directory and then copy the following Python code into the file.

   ```
   import AWSIoTPythonSDK.MQTTLib as AWSIoTPyMQTT

   import json
   import psutil
   import argparse
   import logging
   import time


   # Configures the argument parser for this program.
   def configureParser():
       parser = argparse.ArgumentParser()
       parser.add_argument(
   ```

```
        "-e",
        "--endpoint",
        action="store",
        required=True,
        dest="host",
        help="Your AWS IoT custom endpoint",
    )
    parser.add_argument(
        "-r",
        "--rootCA",
        action="store",
        required=True,
        dest="rootCAPath",
        help="Root CA file path",
    )
    parser.add_argument(
        "-c",
        "--cert",
        action="store",
        required=True,
        dest="certificatePath",
        help="Certificate file path",
    )
    parser.add_argument(
        "-k",
        "--key",
        action="store",
        required=True,
        dest="privateKeyPath",
        help="Private key file path",
    )
    parser.add_argument(
        "-p",
        "--port",
        action="store",
        dest="port",
        type=int,
        default=8883,
        help="Port number override",
    )
    parser.add_argument(
        "-n",
        "--thingName",
        action="store",
```

```
        required=True,
        dest="thingName",
        help="Targeted thing name",
    )
    parser.add_argument(
        "-d",
        "--requestDelay",
        action="store",
        dest="requestDelay",
        type=float,
        default=1,
        help="Time between requests (in seconds)",
    )
    parser.add_argument(
        "-v",
        "--enableLogging",
        action="store_true",
        dest="enableLogging",
        help="Enable logging for the AWS IoT Device SDK for Python",
    )
    return parser


# An MQTT shadow client that uploads device performance data to AWS IoT at a
 regular interval.
class PerformanceShadowClient:
    def __init__(
        self,
        thingName,
        host,
        port,
        rootCAPath,
        privateKeyPath,
        certificatePath,
        requestDelay,
    ):
        self.thingName = thingName
        self.host = host
        self.port = port
        self.rootCAPath = rootCAPath
        self.privateKeyPath = privateKeyPath
        self.certificatePath = certificatePath
        self.requestDelay = requestDelay
```

```
    # Updates this thing's shadow with system performance data at a regular
 interval.
    def run(self):
        print("Connecting MQTT client for {}...".format(self.thingName))
        mqttClient = self.configureMQTTClient()
        mqttClient.connect()
        print("MQTT client for {} connected".format(self.thingName))
        deviceShadowHandler = mqttClient.createShadowHandlerWithName(
            self.thingName, True
        )

        print("Running performance shadow client for {}...
\n".format(self.thingName))
        while True:
            performance = self.readPerformance()
            print("[{}]".format(self.thingName))
            print("CPU:\t{}%".format(performance["cpu"]))
            print("Memory:\t{}%\n".format(performance["memory"]))
            payload = {"state": {"reported": performance}}
            deviceShadowHandler.shadowUpdate(
                json.dumps(payload), self.shadowUpdateCallback, 5
            )
            time.sleep(args.requestDelay)

    # Configures the MQTT shadow client for this thing.
    def configureMQTTClient(self):
        mqttClient = AWSIoTPyMQTT.AWSIoTMQTTShadowClient(self.thingName)
        mqttClient.configureEndpoint(self.host, self.port)
        mqttClient.configureCredentials(
            self.rootCAPath, self.privateKeyPath, self.certificatePath
        )
        mqttClient.configureAutoReconnectBackoffTime(1, 32, 20)
        mqttClient.configureConnectDisconnectTimeout(10)
        mqttClient.configureMQTTOperationTimeout(5)
        return mqttClient

    # Returns the local device's CPU usage, memory usage, and timestamp.
    def readPerformance(self):
        cpu = psutil.cpu_percent()
        memory = psutil.virtual_memory().percent
        timestamp = time.time()
        return {"cpu": cpu, "memory": memory, "timestamp": timestamp}

    # Prints the result of a shadow update call.
```

```
    def shadowUpdateCallback(self, payload, responseStatus, token):
        print("[{}]".format(self.thingName))
        print("Update request {} {}\n".format(token, responseStatus))


# Configures debug logging for the AWS IoT Device SDK for Python.
def configureLogging():
    logger = logging.getLogger("AWSIoTPythonSDK.core")
    logger.setLevel(logging.DEBUG)
    streamHandler = logging.StreamHandler()
    formatter = logging.Formatter(
        "%(asctime)s - %(name)s - %(levelname)s - %(message)s"
    )
    streamHandler.setFormatter(formatter)
    logger.addHandler(streamHandler)


# Runs the performance shadow client with user arguments.
if __name__ == "__main__":
    parser = configureParser()
    args = parser.parse_args()
    if args.enableLogging:
        configureLogging()
    thingClient = PerformanceShadowClient(
        args.thingName,
        args.host,
        args.port,
        args.rootCAPath,
        args.privateKeyPath,
        args.certificatePath,
        args.requestDelay,
    )
    thingClient.run()
```

8. Run `thing_performance.py` from the command line with the following parameters:

   - `-n`, `--thingName` – Your thing name, such as **`SiteWiseTutorialDevice1`**.
   - `-e`, `--endpoint` – Your custom AWS IoT endpoint that you saved earlier in this procedure.
   - `-r`, `--rootCA` – The path to your AWS IoT root CA certificate.
   - `-c`, `--cert` – The path to your AWS IoT thing certificate.
   - `-k`, `--key` – The path to your AWS IoT thing certificate private key.

- `-d, --requestDelay` – (Optional) The time in seconds to wait between each device shadow update. Defaults to 1 second.
- `-v, --enableLogging` – (Optional) If this parameter is present, the script prints debug messages from the AWS IoT Device SDK for Python.

Your command should look similar to the following example.

```
python3 thing_performance.py \
  --thingName SiteWiseTutorialDevice1 \
  --endpoint identifier.iot.region.amazonaws.com \
  --rootCA AmazonRootCA1.pem \
  --cert device1/thing-id-certificate.pem.crt \
  --key device1/thing-id-private.pem.key
```

If you're running the script for additional AWS IoT things, update the thing name and certificate directory accordingly.

9. Try opening and closing programs on your device to see how the CPU and memory usages change. The script prints each CPU and memory usage reading. If the script uploads data to the device shadow service successfully, the script's output should look like the following example.

```
[SiteWiseTutorialDevice1]
CPU:    24.6%
Memory: 85.2%

[SiteWiseTutorialDevice1]
Update request e6686e44-fca0-44db-aa48-3ca81726f3e3 accepted
```

10. Follow these steps to verify that the script is updating the device shadow:

    a. Navigate to the AWS IoT console.

    b. In the left navigation pane, choose **All devices** and then choose **Things**.

    c. Choose your thing, **SiteWiseTutorialDevice**.

    d. Choose the **Device Shadows** tab, choose **Classic Shadow**, and verify that the **Shadow state** looks like the following example.

```
{
  "reported": {
```

```
    "cpu": 24.6,
    "memory": 85.2,
    "timestamp": 1579567542.2835066
  }
}
```

If your thing's shadow state is empty or doesn't look like the previous example, check that the script is running and successfully connected to AWS IoT. If the script continues to time out when connecting to AWS IoT, check that your thing policy is configured according to this tutorial.

11. Follow these steps to verify that the rule action is sending data to AWS IoT SiteWise:

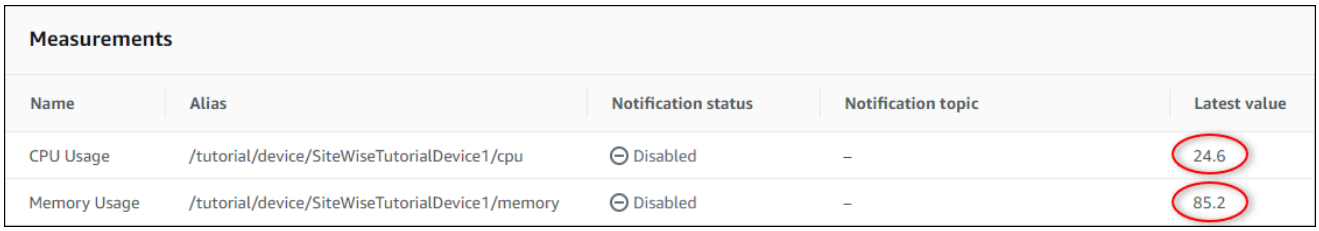
    a. Navigate to the AWS IoT SiteWise console.

    b. In the left navigation pane, choose **Assets**.

    c. Choose the arrow next to your device fleet asset (**SiteWise Tutorial Device Fleet 1 1**) to expand its asset hierarchy, and then choose your device asset (**SiteWise Tutorial Device 1**).

    d. Choose **Measurements**.

    e. Verify that the **Latest value** cells have values for the **CPU Usage** and **Memory Usage** properties.



    f. If the **CPU Usage** and **Memory Usage** properties don't have the latest values, refresh the page. If values don't appear after a few minutes, see Troubleshooting a rule.

12. You have completed this tutorial. If you want to explore live visualizations of your data, you can configure a portal in AWS IoT SiteWise Monitor. For more information, see Monitoring data with AWS IoT SiteWise Monitor. Otherwise, you can press **CTRL+C** in your command prompt to stop the device client script. It's unlikely the Python program will send enough messages to incur charges, but it's a best practice to stop the program when you're done.

Step 9: Cleaning up resources after the tutorial

After you complete the tutorial about ingesting data from AWS IoT things, clean up your resources to avoid incurring additional charges.

To delete hierarchical assets in AWS IoT SiteWise

1. Navigate to the AWS IoT SiteWise console
2. In the left navigation pane, choose **Assets**.
3. When you delete assets in AWS IoT SiteWise, you must first disassociate them.

   Complete the following steps to disassociate your device assets from your device fleet asset:

   a. Choose your device fleet asset (**SiteWise Tutorial Device Fleet 1**).

   b. Choose **Edit**.

   c. Under **Assets associated to this asset**, choose **Disassociate** for each device asset associated to this device fleet asset.

   d. Choose **Save**.

   You should now see your device assets no longer organized as a hierarchy.
4. Choose your device asset (**SiteWise Tutorial Device 1**).
5. Choose **Delete**.
6. In the confirmation dialog, enter **`Delete`** and then choose **Delete**.
7. Repeat steps 4 through 6 for each device asset and the device fleet asset (**SiteWise Tutorial Device Fleet 1**).

To delete hierarchical asset models in AWS IoT SiteWise

1. Navigate to the AWS IoT SiteWise console.
2. If you haven't already, delete your device and device fleet assets. For more information, see the previous procedure. You can't delete a model if you have assets that were created from that model.
3. In the left navigation pane, choose **Models**.
4. Choose your device fleet asset model (**SiteWise Tutorial Device Fleet Model**).

   When deleting hierarchical asset models, start by deleting the parent asset model first.
5. Choose **Delete**.
6. In the confirmation dialog, enter **`Delete`** and then choose **Delete**.
7. Repeat steps 4 through 6 for your device asset model (**SiteWise Tutorial Device Model**).

To disable or delete a rule in AWS IoT Core

1. Navigate to the AWS IoT console.
2. In the left navigation pane, choose **Message routing** and then choose **Rules**.
3. Select your rule and choose **Delete**.
4. In the confirmation dialog, enter the name of the rule and then choose **Delete**.

Visualizing and sharing wind farm data in SiteWise Monitor

This tutorial explains how to use AWS IoT SiteWise Monitor to visualize and share industrial data through managed web applications, known as portals. Each *portal* encompasses projects, providing you with the flexibility to choose which data is accessible within each project. Then, specify people in your organization that can access each portal. Your users sign in to portals using AWS IAM Identity Center accounts, so you can use your existing identity store or a store managed by AWS.

You, and your users with sufficient permissions, can create dashboards in each project to visualize your industrial data in meaningful ways. Then, your users can view these dashboards to quickly gain insights into your data and monitor your operation. You can configure administrative or read-only permissions to each project for every user in your company. For more information, see Monitoring data with AWS IoT SiteWise Monitor.

Throughout the tutorial, you enhance the AWS IoT SiteWise demo, providing a sample dataset for a wind farm. You configure a portal in SiteWise Monitor, create a project, and dashboards to visualize the wind farm data. The tutorial also covers the creation of additional users, along with the assignment of permissions to own or view the project and its associated dashboards.

> **Note**
>
> When you use SiteWise Monitor, you're charged per user that signs in to a portal (per month). In this tutorial, you create three users, but you only need to sign in with one user. After you complete this tutorial, you incur charges for one user. For more information, see AWS IoT SiteWise Pricing.

Topics

Prerequisites

To complete this tutorial, you need the following:

- An AWS account. If you don't have one, see Setting up an AWS account.
- A development computer running Windows, macOS, Linux, or Unix to access the AWS Management Console. For more information, see Getting Started with the AWS Management Console.
- An AWS Identity and Access Management (IAM) user with administrator permissions.
- A running AWS IoT SiteWise wind farm demo. When you set up the demo, it defines models and assets in AWS IoT SiteWise and streams data to them to represent a wind farm. For more information, see Using the AWS IoT SiteWise demo.
- If you enabled IAM Identity Center in your account, sign in to your AWS Organizations management account. For more information, see AWS Organizations terminology and concepts. If you haven't enabled IAM Identity Center, you will enable it in this tutorial and set your account as the management account.

  If you can't sign in to your AWS Organizations management account, you can partially complete the tutorial as long as you have an IAM Identity Center user in your organization. In this case, you can create the portal and dashboards, but you can't create new IAM Identity Center users to assign to projects.

Step 1: Create a portal in SiteWise Monitor

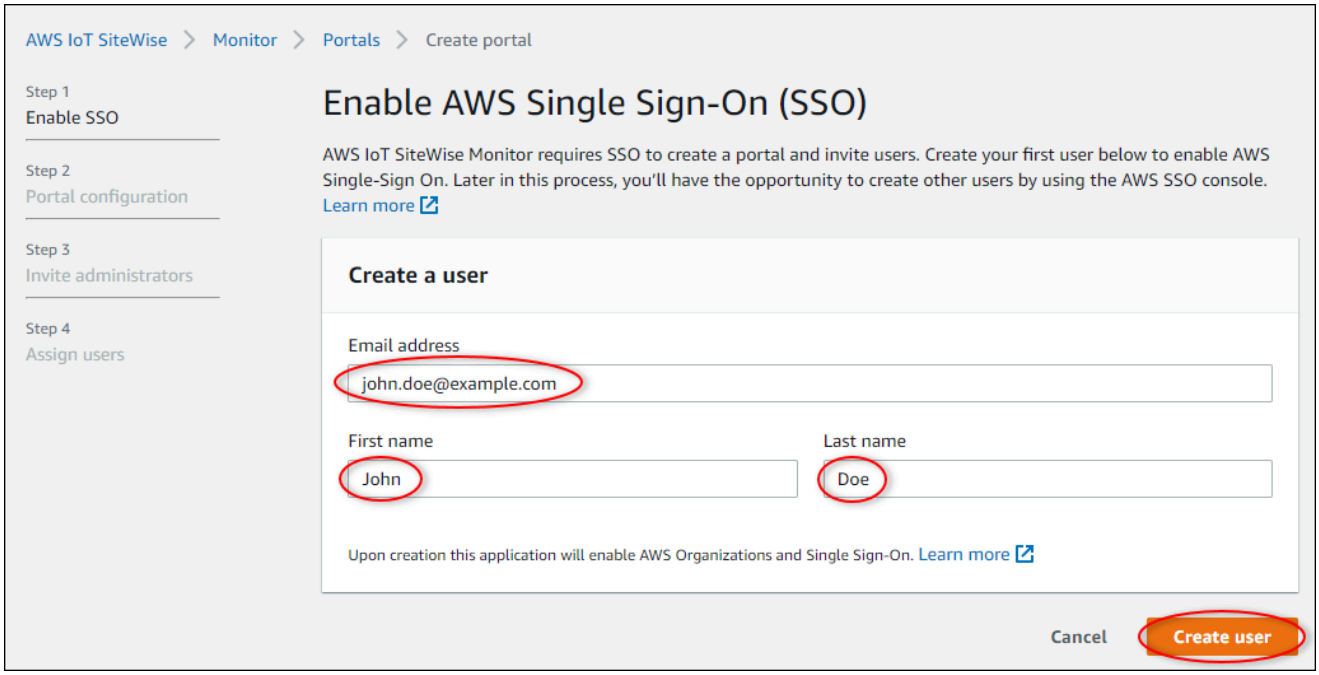
In this procedure, you create a portal in AWS IoT SiteWise Monitor. Each *portal* is a managed web application that you and your users can sign in to with AWS IAM Identity Center accounts. With IAM Identity Center, you can use your company's existing identity store or create one managed by AWS. Your company's employees can sign in without creating separate AWS accounts.

To create a portal

1. Sign in to the AWS IoT SiteWise console.
2. Review the AWS IoT SiteWise endpoints and quotas where AWS IoT SiteWise is supported and switch Regions, if needed. You must run the AWS IoT SiteWise demo in the same Region.
3. In the left navigation pane, choose **Portals**.
4. Choose **Create portal**.
5. If you already enabled IAM Identity Center, skip to step 6. Otherwise, complete the following steps to enable IAM Identity Center:

   a. On the **Enable AWS IAM Identity Center (SSO)** page, enter your **Email address**, **First name**, and **Last name** to create an IAM Identity Center user for yourself to be the portal administrator. Use an email address you can access so that you can receive an email to set a password for your new IAM Identity Center user.

   In a portal, the portal administrator creates projects and assigns users to projects. You can create more users later.



   b. Choose **Create user**.

6. On the **Portal configuration** page, complete the following steps:

   a. Enter a name for your portal, such as **`WindFarmPortal`**.

b. (Optional) Enter a description for your portal. If you have multiple portals, use meaningful descriptions to keep track of what each portal contains.

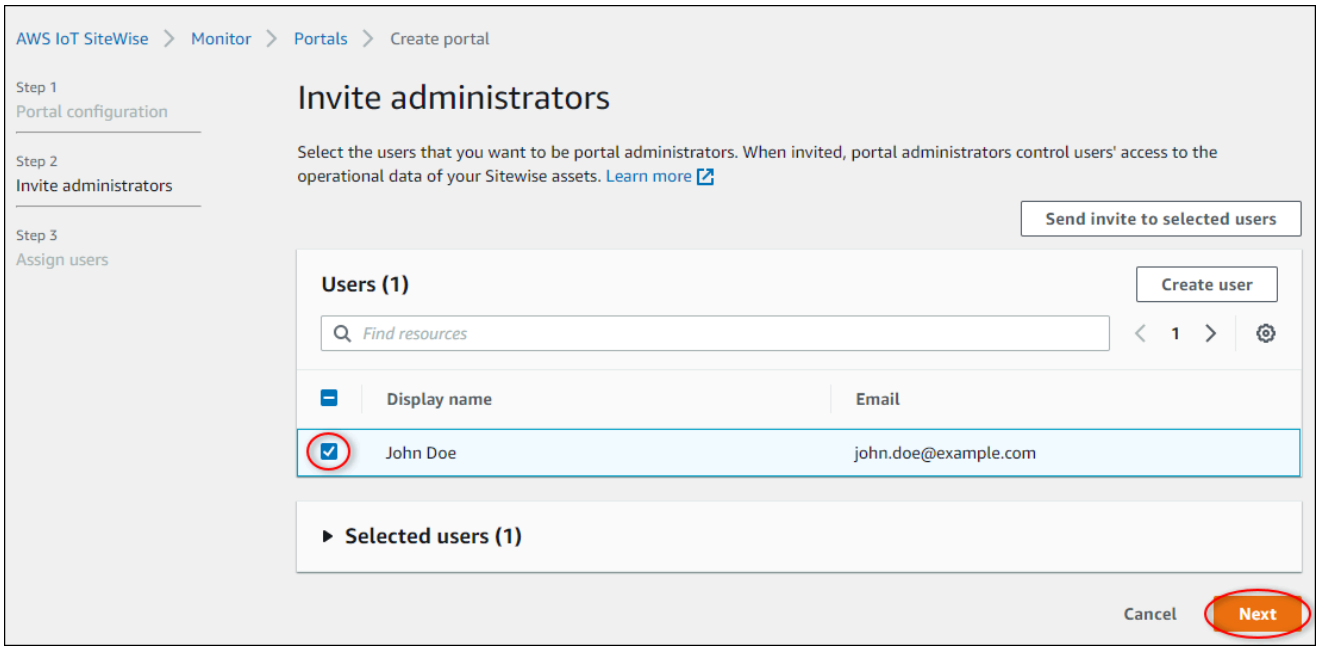
c. (Optional) Upload an image to display in the portal.

d. Enter an email address that portal users can contact when they have an issue with the portal and need help from your company's AWS administrator to resolve it.

e. Choose **Create portal**.

7. On the **Invite administrators** page, you can assign IAM Identity Center users to the portal as administrators. Portal administrators manage permissions and projects within a portal. On this page, do the following:

a. Select a user to be the portal administrator. If you enabled IAM Identity Center earlier in this tutorial, select the user that you created.



b. (Optional) Choose **Send invite to selected users**. Your email client opens, and an invitation appears in the message body. You can customize the email before you send it to your portal administrators. You can also send the email to your portal administrators later. If you're trying SiteWise Monitor for the first time and will be the portal administrator, you don't need to email yourself.

c. Choose **Next**.

8. On the **Assign users** page, you can assign IAM Identity Center users to the portal. Portal administrators can later assign these users as project owners or viewers. Project owners can

create dashboards in projects. Project viewers have read-only access to the projects that they're assigned. On this page, you can create IAM Identity Center users to add to the portal.

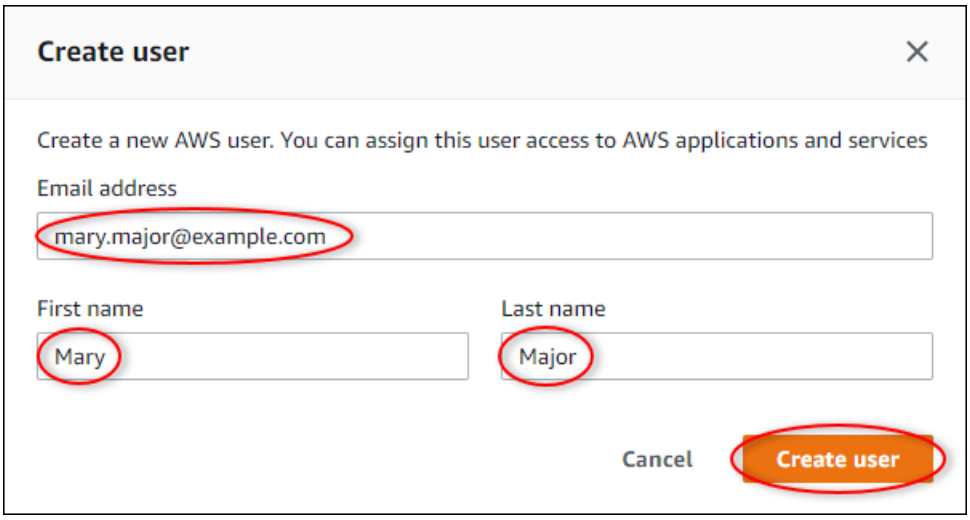
> **Note**
>
> If you aren't signed in to your AWS Organizations management account, you can't create IAM Identity Center users. Choose **Assign users** to create the portal without portal users, and then skip this step.

On this page, do the following:

a. Complete the following steps twice to create two IAM Identity Center users:

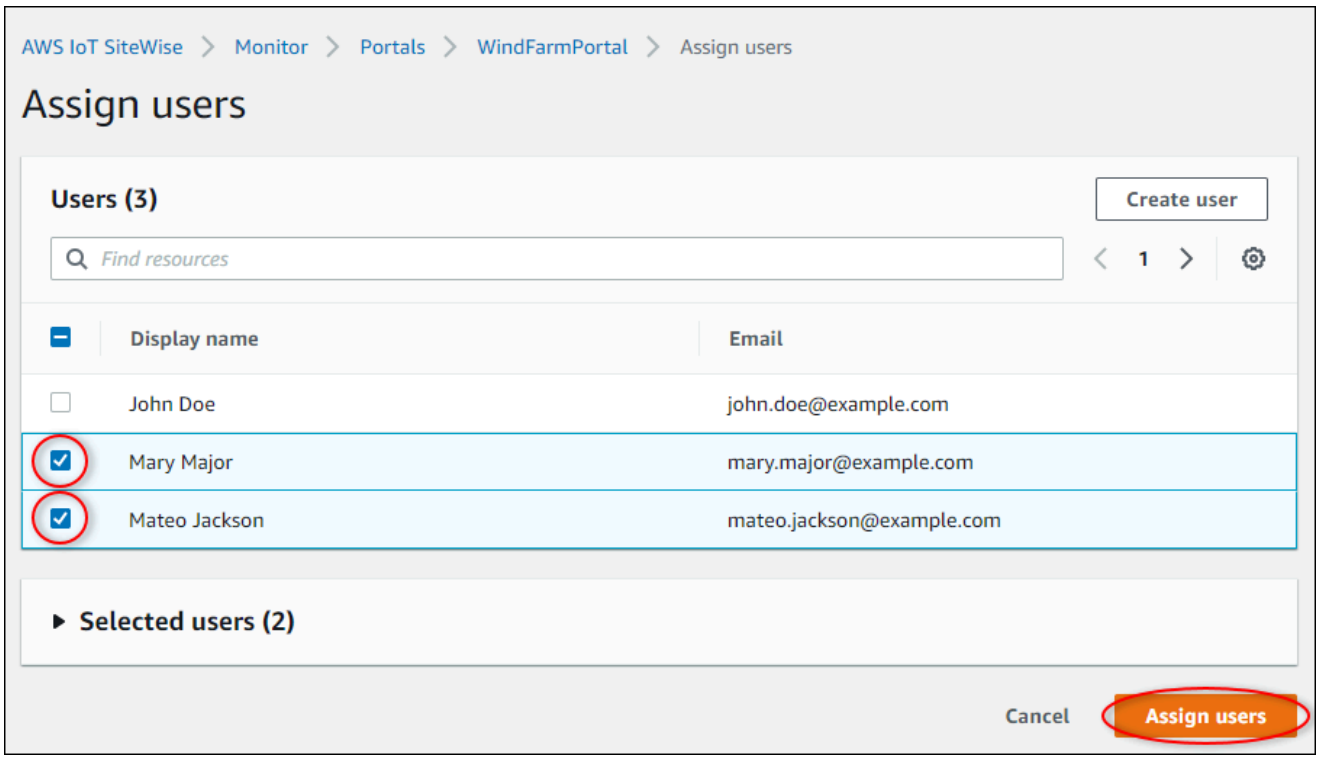
   i. Choose **Create user** to open a dialog box where you enter details for the new user.

   ii. Enter an **Email address**, **First name**, and **Last name** for the new user. IAM Identity Center sends the user an email for them to set their password. If you want to sign in to the portal as these users, choose an email address that you can access. Each email address must be unique. Your users sign in to the portal using their email address as their usernames.



   iii. Choose **Create user**.

b. Select the two IAM Identity Center users that you created in the previous step.



c. Choose **Assign users** to add these users to the portal.

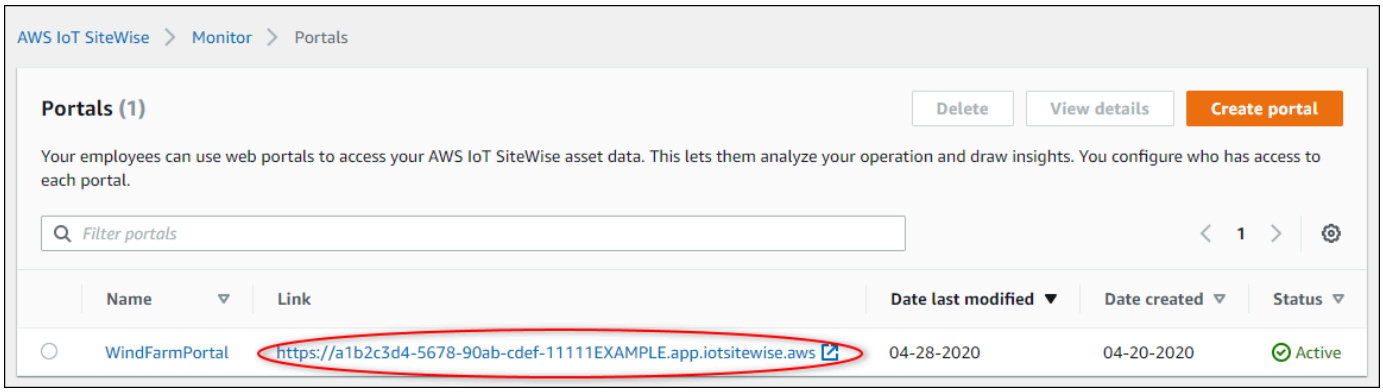
The portals page opens with your new portal listed.

Step 2: Sign in to a portal

In this procedure, you sign in to your new portal using the AWS IAM Identity Center user that you added to the portal.

To sign in to a portal

1. On the **Portals** page, choose your new portal's **Link** to open your portal in a new tab.



2. If you created your first IAM Identity Center user earlier in the tutorial, use the following steps to create a password for your user:

   a. Check your email for the subject line **Invitation to join AWS IAM Identity Center**.

   b. Open that invitation email and choose **Accept invitation**.

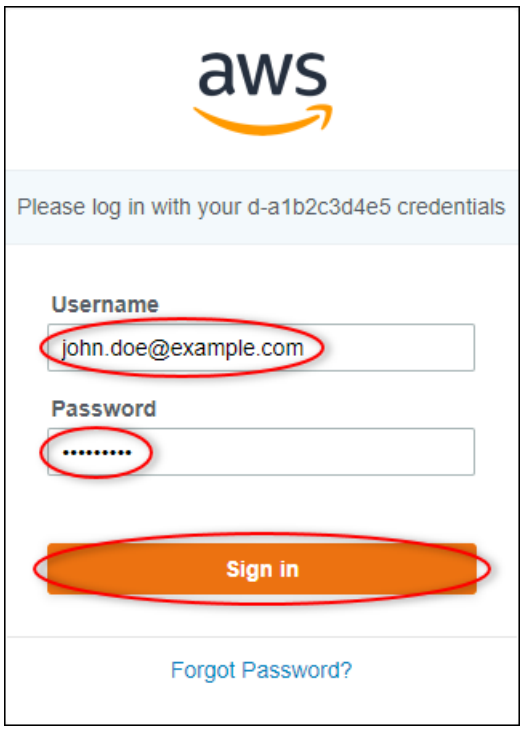
   c. In the new window, set a password for your IAM Identity Center user.

   If you want to sign in later to the portal as the second and third IAM Identity Center users that you created earlier, you can also complete these steps to set passwords for those users.

   > **Note**
   >
   > If you didn't receive an email, you can generate a password for your user in the IAM Identity Center console. For more information, see Reset a user password in the *AWS IAM Identity Center User Guide*.

3. Enter your IAM Identity Center **Username** and **Password**. If you created your IAM Identity Center user earlier in this tutorial, your **Username** is the email address of the portal administrator user that you created.

   All portal users, including the portal administrator, must sign in with their IAM Identity Center user credentials. These credentials are typically not the same credentials that you use to sign in to the AWS Management Console.



4. Choose **Sign in**.

   Your portal opens.

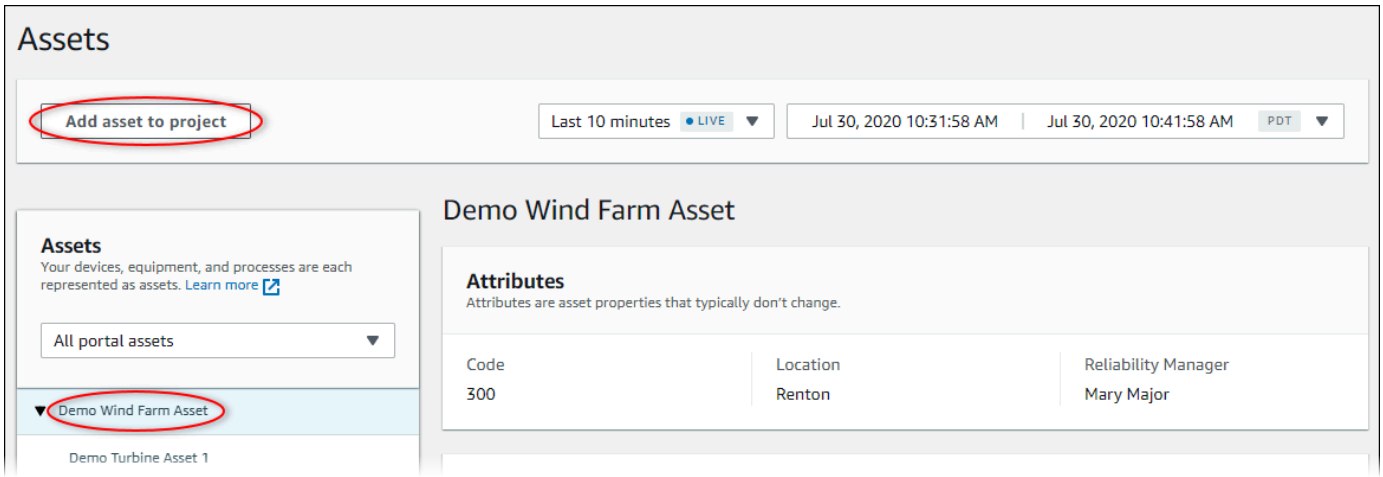
Step 3: Create a wind farm project

In this procedure, you create a project in your portal. *Projects* are resources that define a set of permissions, assets, and dashboards, which you can configure to visualize asset data in that project. With projects, you define who has access to which subsets of your operation and how those subsets' data is visualized. You can assign portal users as owners or viewers of each project. Project owners can create dashboards to visualize data and share the project with other users. Project viewers can view dashboards but not edit them. For more information about roles in SiteWise Monitor, see SiteWise Monitor roles.

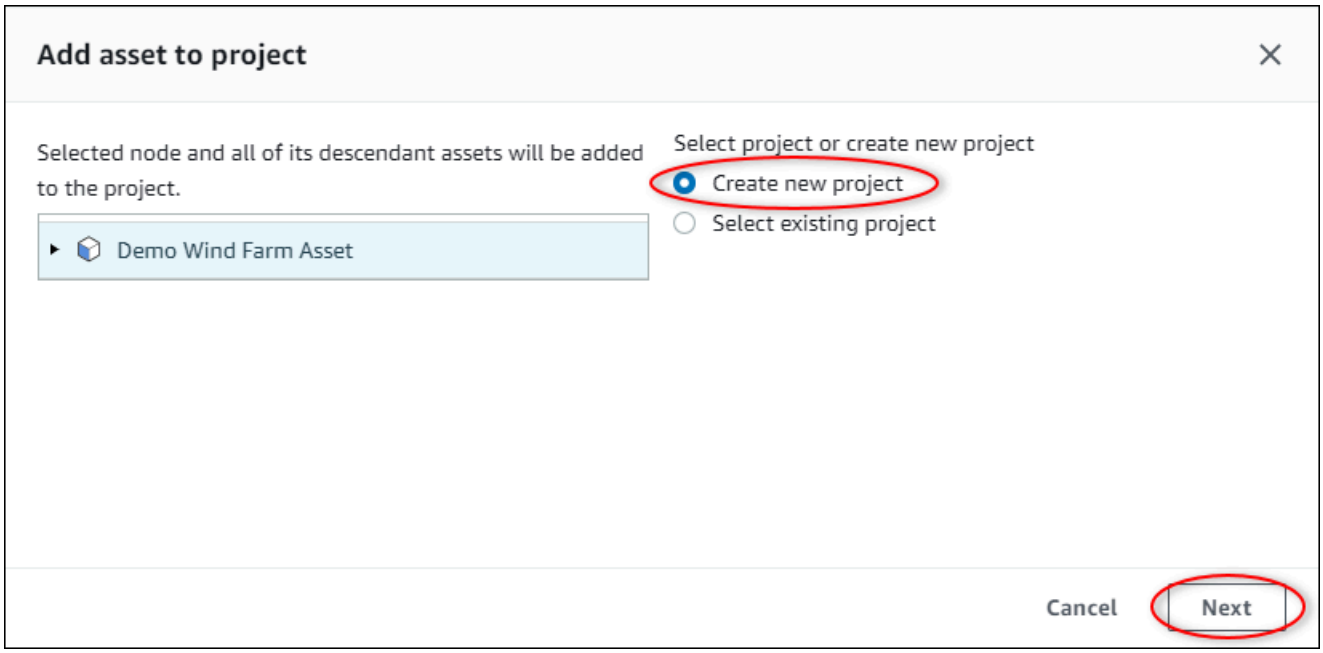
To create a wind farm project

1. In the left navigation pane in your portal, choose the **Assets** tab. On the **Assets** page, you can explore all assets available in the portal and add assets to projects.
2. In the asset browser, choose **Demo Wind Farm Asset**. When you choose an asset, you can explore that asset's live and historical data. You can also press **Shift** to select multiple assets and compare their data side-by-side.

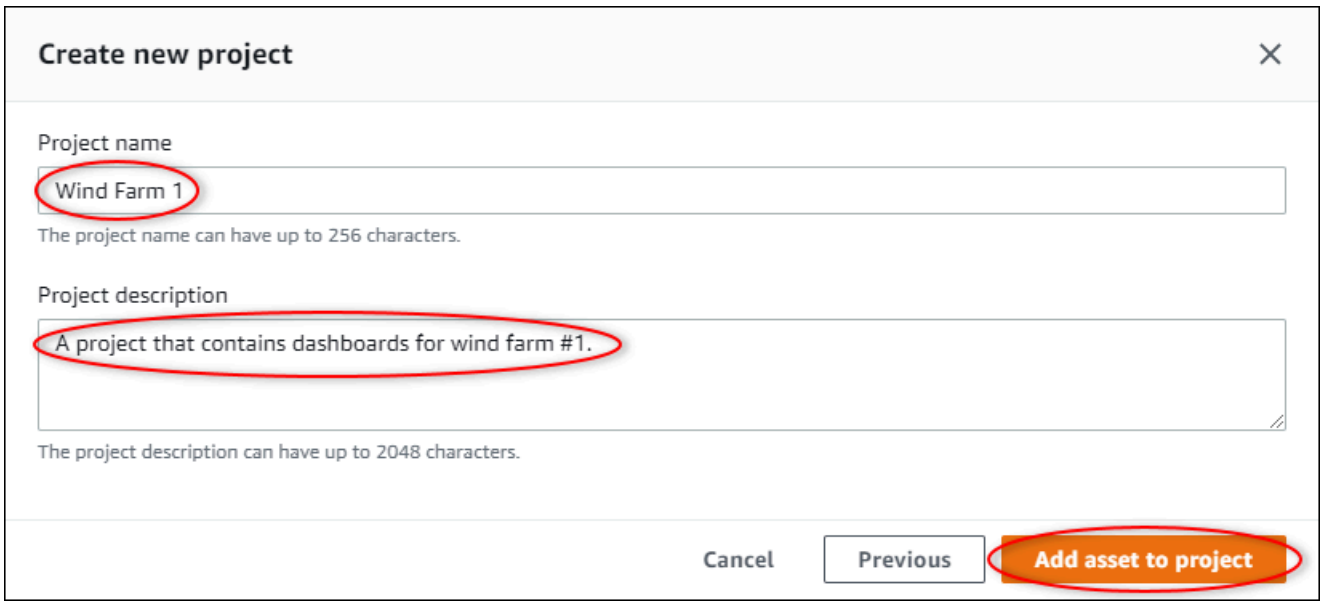
3. Choose **Add asset to project** in the upper left. Projects contain dashboards that your portal users can view to explore your data. Each project has access to a subset of your assets in AWS IoT SiteWise. When you add an asset to a project, all users with access to that project can also access data for that asset and its children.



4. In the **Add asset to project** dialog box, choose **Create new project**, and then choose **Next**.



5. In the **Create new project** dialog box, enter a **Project name** and **Project description** for your project, and then choose **Add asset to project**.



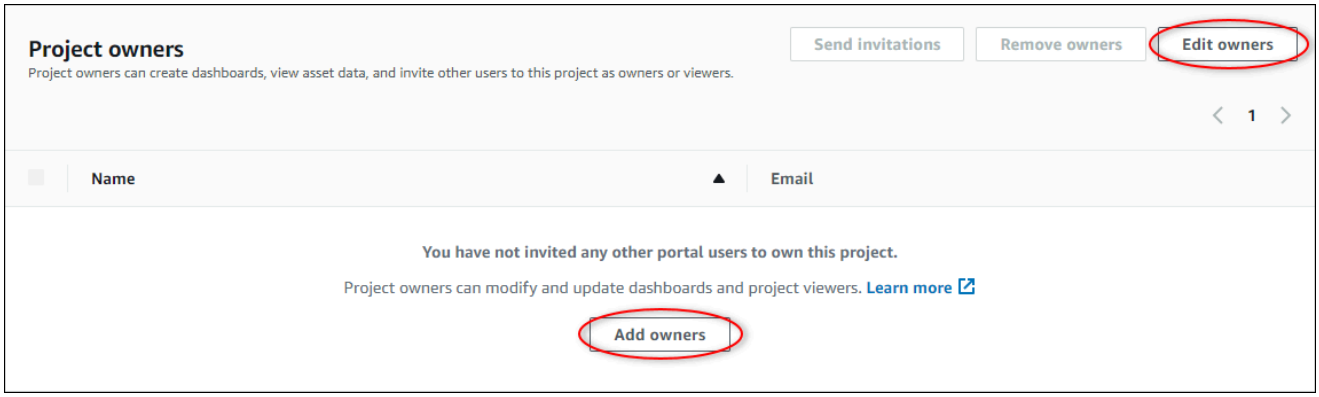
Your new project's page opens.

6. On the project's page, you can add portal users as owners or viewers of this project.

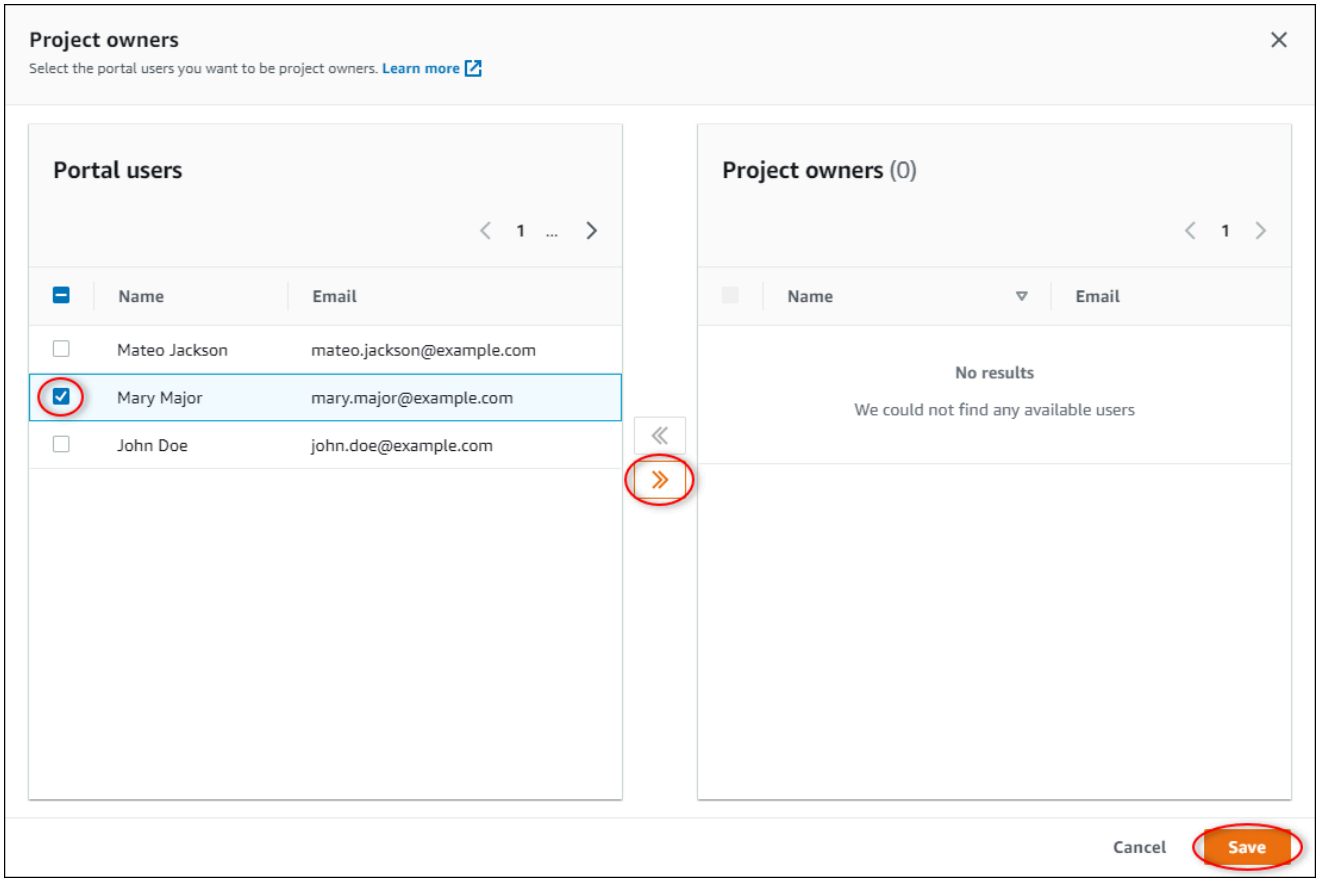
> **Note**
>
> If you aren't signed in to your AWS Organizations management account, you might not have portal users to assign to this project, so you can skip this step.

On this page, do the following:

a. Under **Project owners**, choose **Add owners** or **Edit users**.



b. Choose the user to add as a project owner (for example, **Mary Major**), and then choose the **>>** icon.



c. Choose **Save**.

   Your IAM Identity Center user **Mary Major** can sign in to this portal to edit the dashboards in this project and share this project with other users in this portal.

d. Under **Project viewers**, choose **Add viewers** or **Edit users**.

e. Choose the user to add as a project viewer (for example, **Mateo Jackson**), and then choose the **>>** icon.

f. Choose **Save**.

   Your IAM Identity Center user **Mateo Jackson** can sign in to this portal to view, but not edit, the dashboards in the wind farm project.

Step 4: Create a dashboard to visualize wind farm data

In this procedure, you create dashboards to visualize the demo wind farm data. Dashboards contain customizable visualizations of your project's asset data. Each visualization can have a different type, such as a line chart, bar chart, or key performance indicator (KPI) display. You can choose

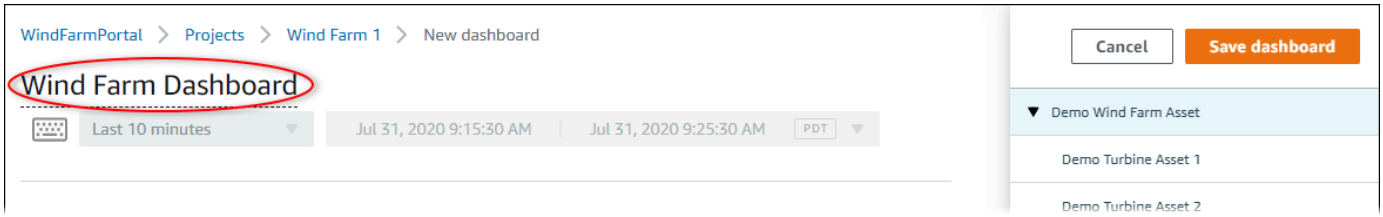
the visualization type that works best for your data. Project owners can edit dashboards, whereas project viewers can only view dashboards to gain insights.

To create a dashboard with visualizations

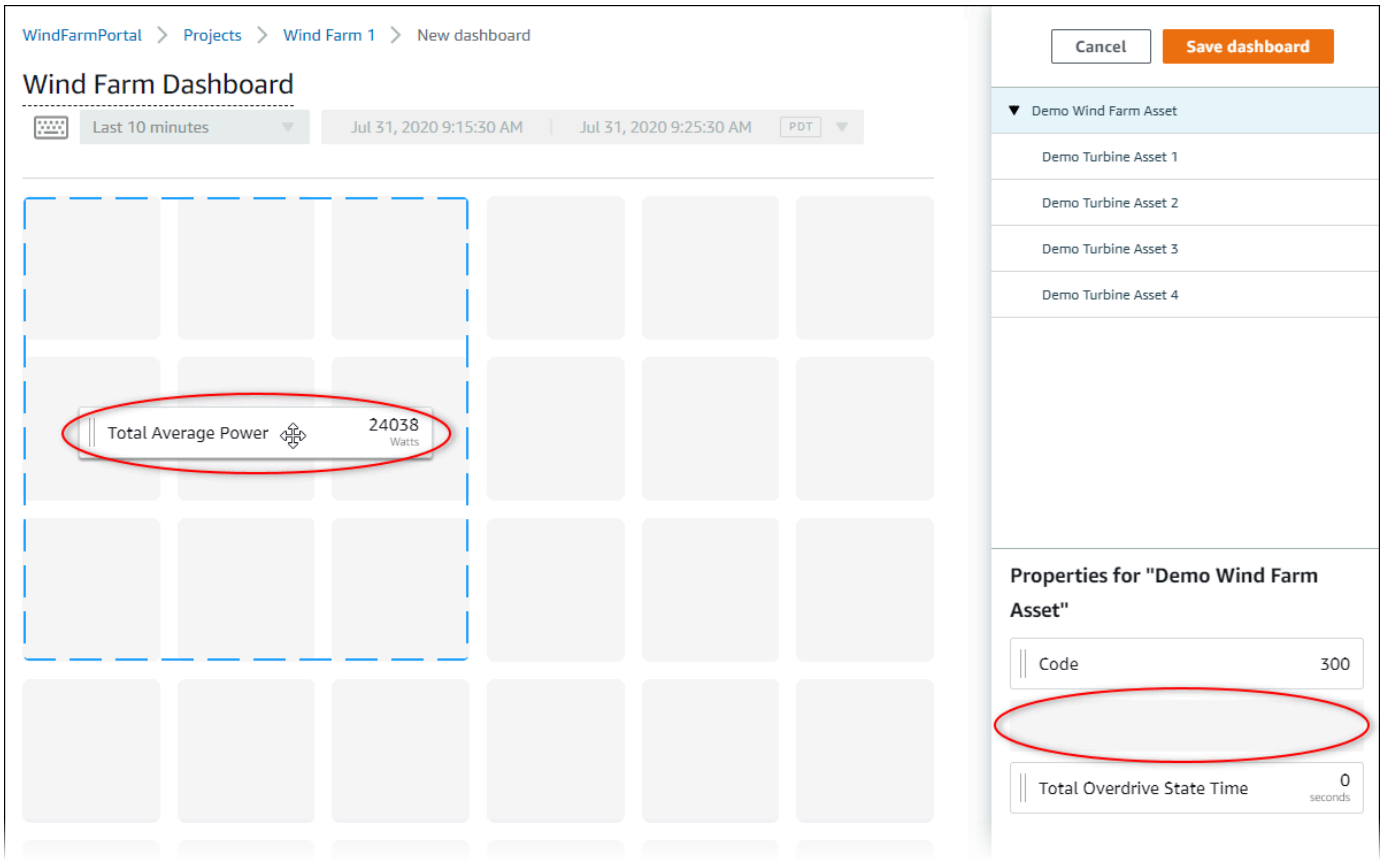
1. On your new project's page, choose **Create dashboard** to create a dashboard and open its edit page.

   In a dashboard's edit page, you can drag asset properties from the asset hierarchy to the dashboard to create visualizations. Then, you can edit each visualization's title, legend titles, type, size, and location in the dashboard.

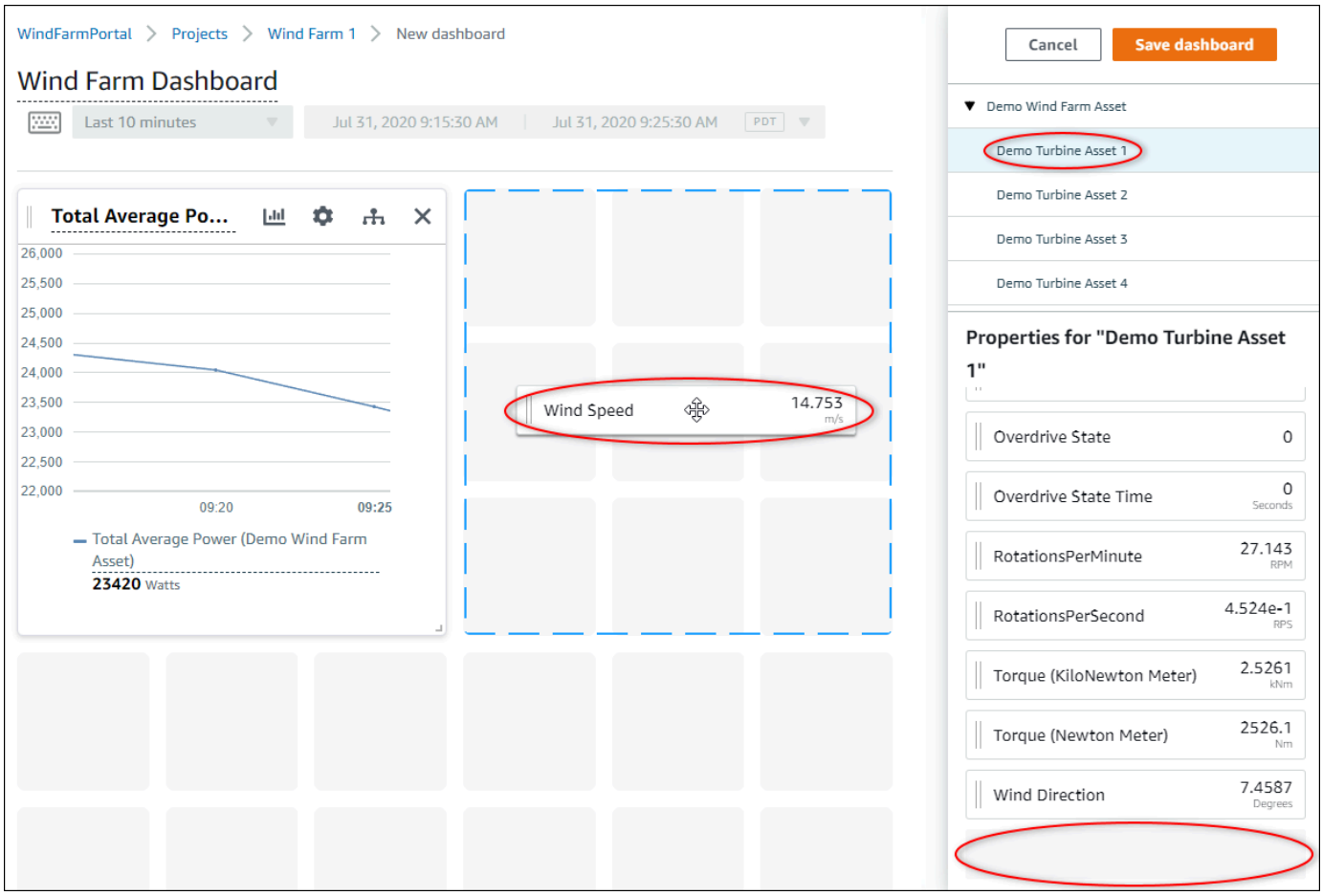
2. Enter a name your dashboard.



3. Drag **Total Average Power** from the **Demo Wind Farm Asset** to the dashboard to create a visualization.

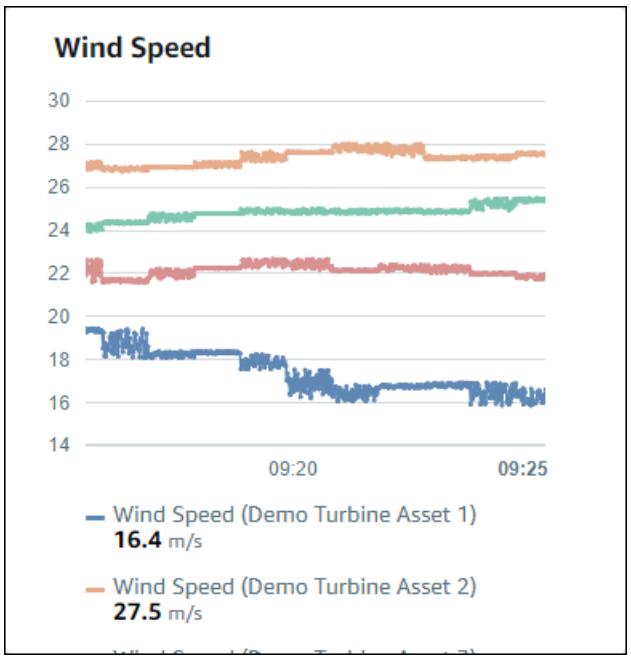


4. Choose **Demo Turbine Asset 1** to show properties for that asset, and then drag **Wind Speed** to the dashboard to create a visualization for wind speed.

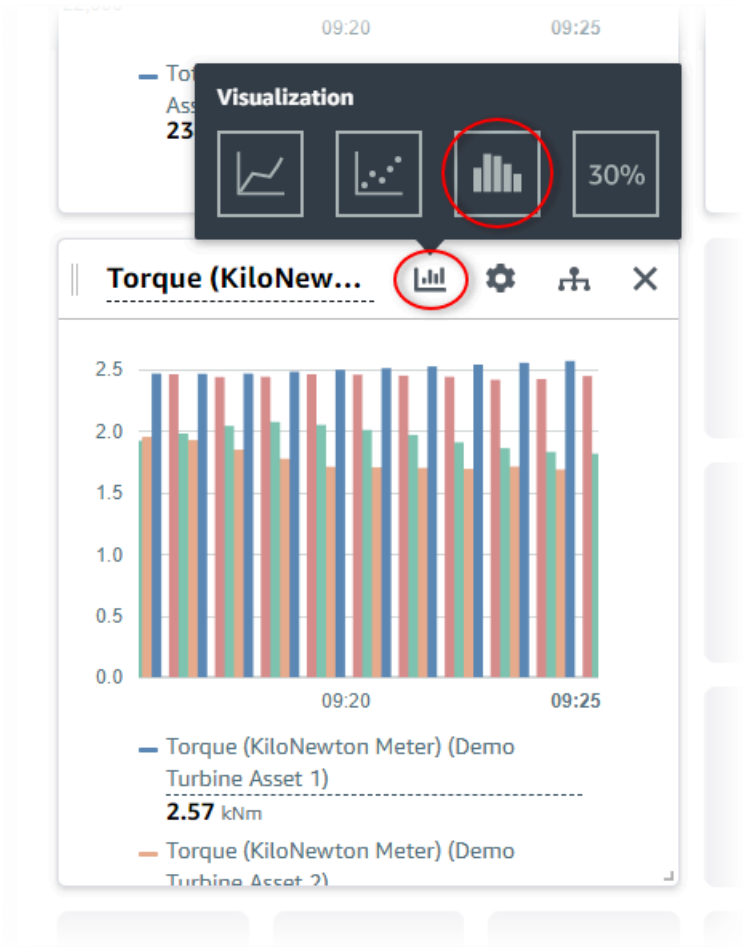


5. Add **Wind Speed** to the new wind speed visualization for each **Demo Turbine Asset 2**, **3**, and **4** (in that order).

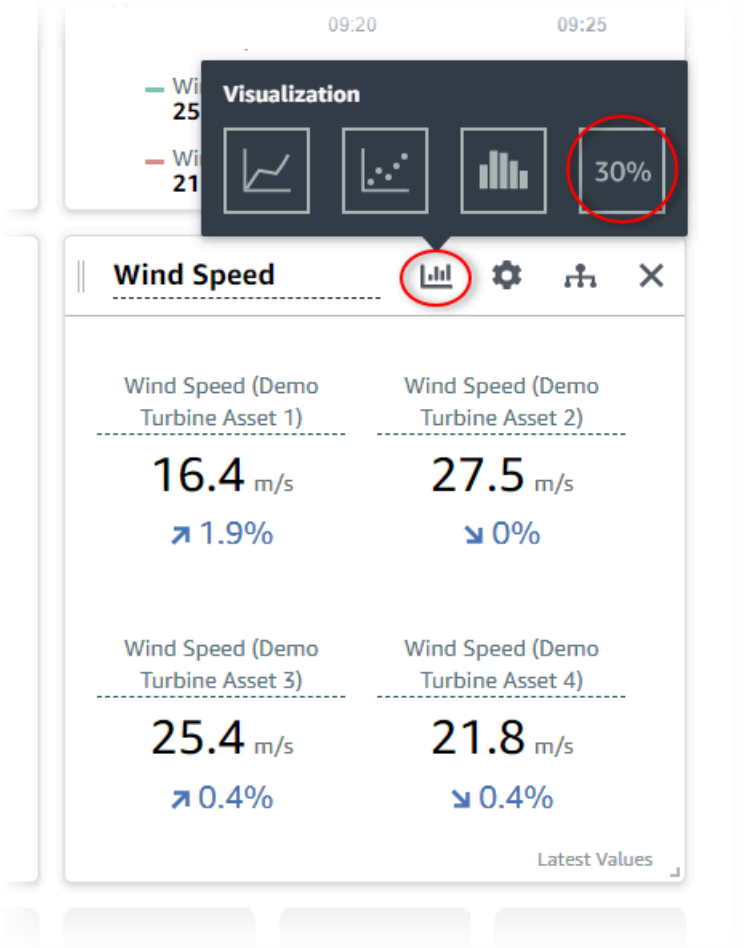
   Your **Wind Speed** visualization should look similar to the following screenshot.



6. Repeat steps 4 and 5 for the wind turbines' **Torque (KiloNewton Meter)** properties to create a visualization for wind turbine torque.
7. Choose the visualization type icon for the **Torque (KiloNewton Meter)** visualization, and then choose the bar chart icon.

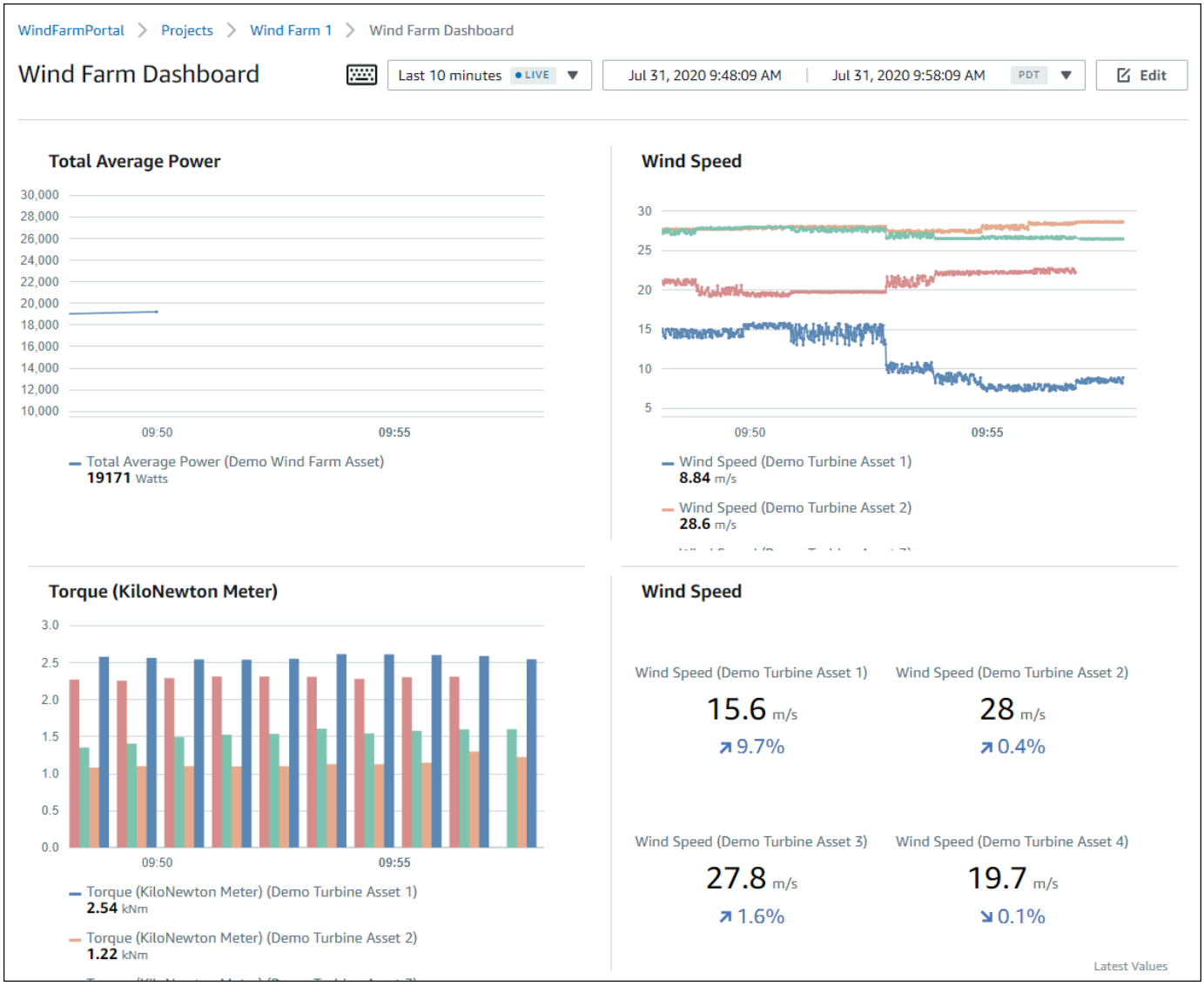


8. Repeat steps 4 and 5 for the wind turbines' **Wind Direction** properties to create a visualization for wind direction.
9. Choose the visualization type icon for the **Wind Direction** visualization, and then choose the KPI chart icon (**30%**).



10. (Optional) Make other changes to each visualization's title, legend titles, type, size, and location as needed.
11. Choose **Save dashboard** in the upper right to save your dashboard.

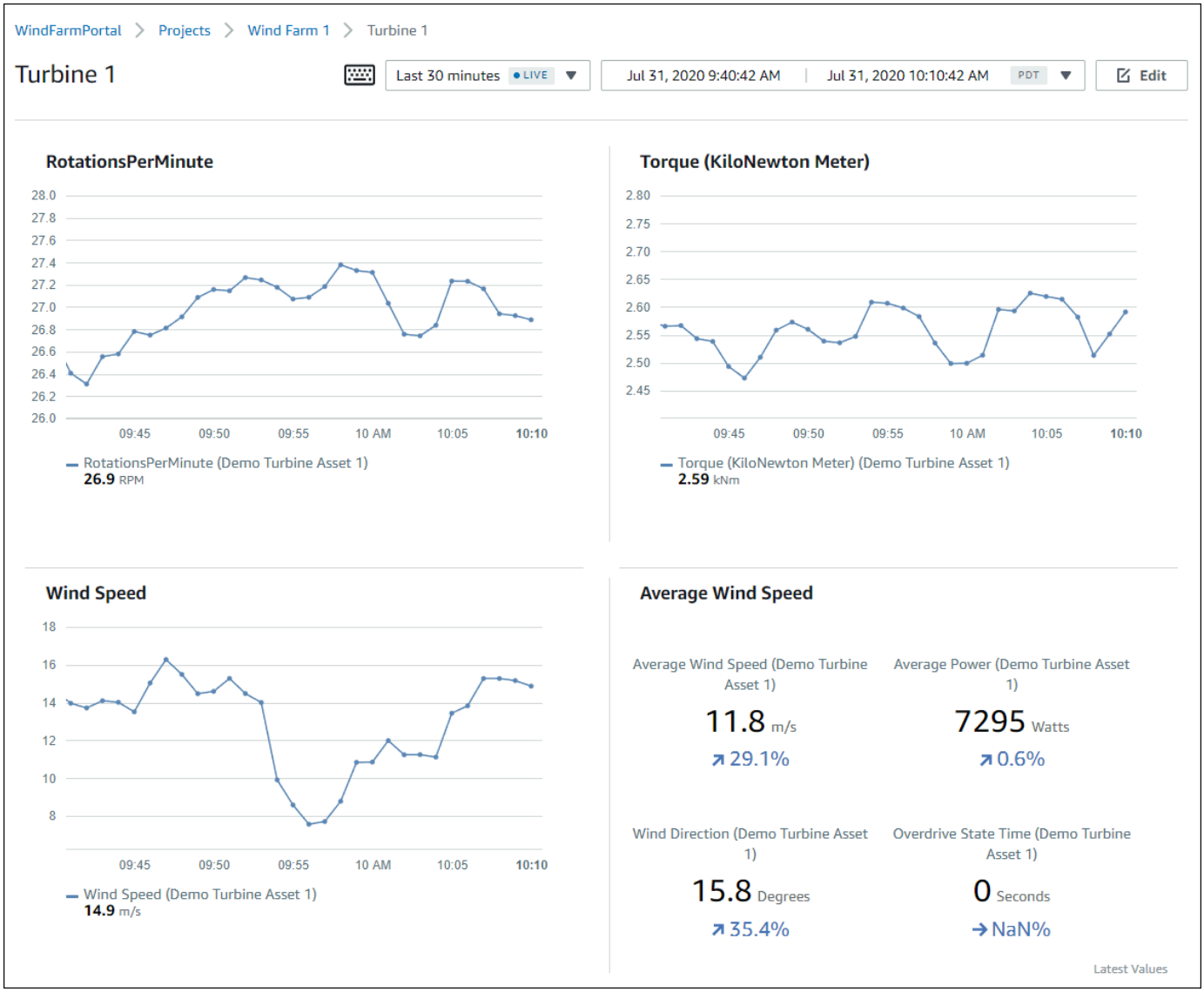
    Your dashboard should look similar to the following screenshot.



12. (Optional) Create an additional dashboard for each wind turbine asset.

    As a best practice, we recommend that you create a dashboard for each asset so that your project viewers can investigate any issues with each individual asset. You can only add up to 5 assets to each visualization, so you must create multiple dashboards for your hierarchical assets in many scenarios.

    A dashboard for a demo wind turbine might look similar to the following screenshot.



13. (Optional) Change the timeline or select data points on a visualization to explore the data in your dashboard. For more information, see Viewing dashboards in the *AWS IoT SiteWise Monitor Application Guide*.

Step 5: Explore the portal

In this procedure, you can explore the portal as a user with fewer permissions than an AWS IoT SiteWise portal administrator.

To explore the portal and finish the tutorial

- (Optional) If you added other users to the project as owners or viewers, you can sign in to the portal as these users. This lets you explore the portal as a user with fewer permissions than a portal administrator.

  > **Important**
  >
  > You're charged for each user that signs in to a portal. For more information, see AWS IoT SiteWise Pricing.

  To explore the portal as other users, do the following:

  a. Choose **Log out** in the bottom left of the portal to exit the web application.
  b. Choose **Sign out** in the upper right of the IAM Identity Center application portal to sign out of your IAM Identity Center user.
  c. Sign in to the portal as the IAM Identity Center user that you assigned as a project owner or project viewer. For more information, see Step 2: Sign in to a portal.

You've completed the tutorial. When you finish exploring your demo wind farm in SiteWise Monitor, follow the next procedure to clean up your resources.

Step 6: Clean up resources after the tutorial

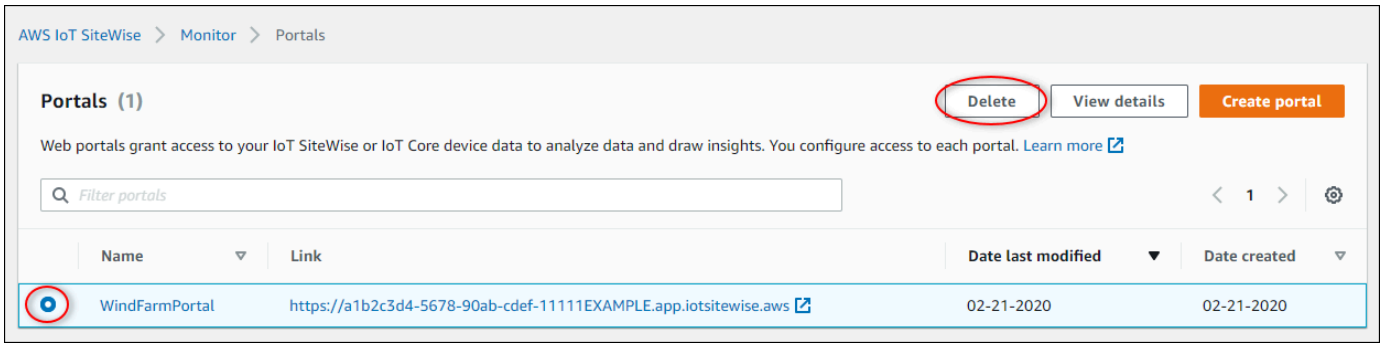
After you complete the tutorial, you can clean up your resources. You aren't charged for AWS IoT SiteWise if users don't sign in to your portal, but you can delete your portal and AWS IAM Identity Center directory users. Your demo wind farm assets are deleted at the end of the duration that you chose when you created the demo, or you can delete the demo manually. For more information, see Deleting the AWS IoT SiteWise demo.

Use the following procedures to delete your portal and IAM Identity Center users.

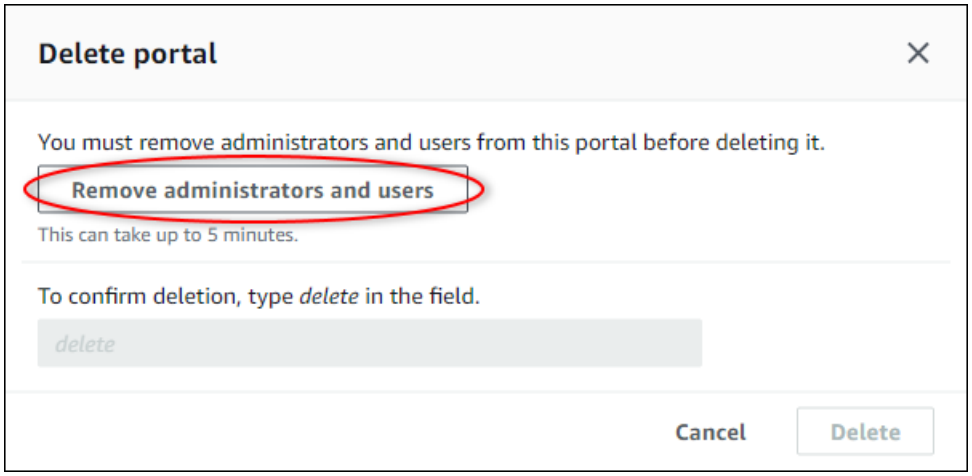
To delete a portal

1. Navigate to the AWS IoT SiteWise console.
2. In the left navigation pane, choose **Portals**.
3. Choose your portal, **WindFarmPortal**, and then choose **Delete**.

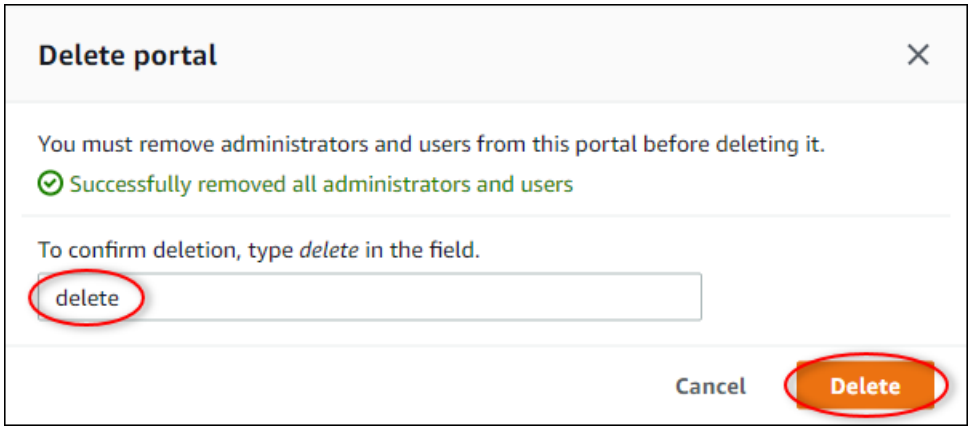
When you delete a portal or project, the assets associated to deleted projects aren't affected.



4. In the **Delete portal** dialog box, choose **Remove administrators and users**.



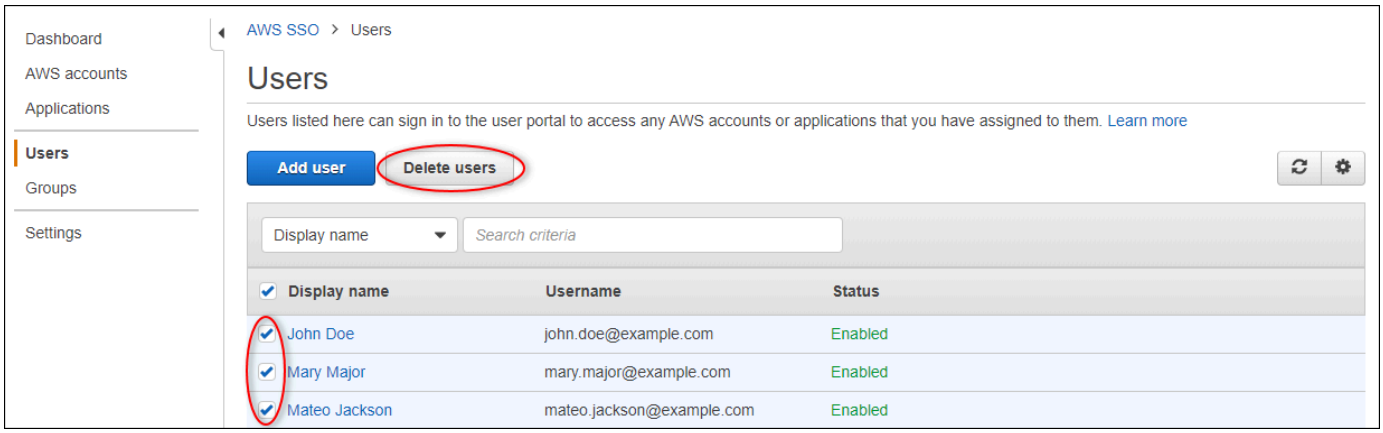
5. Enter **`delete`** to confirm deletion, and then choose **Delete**.



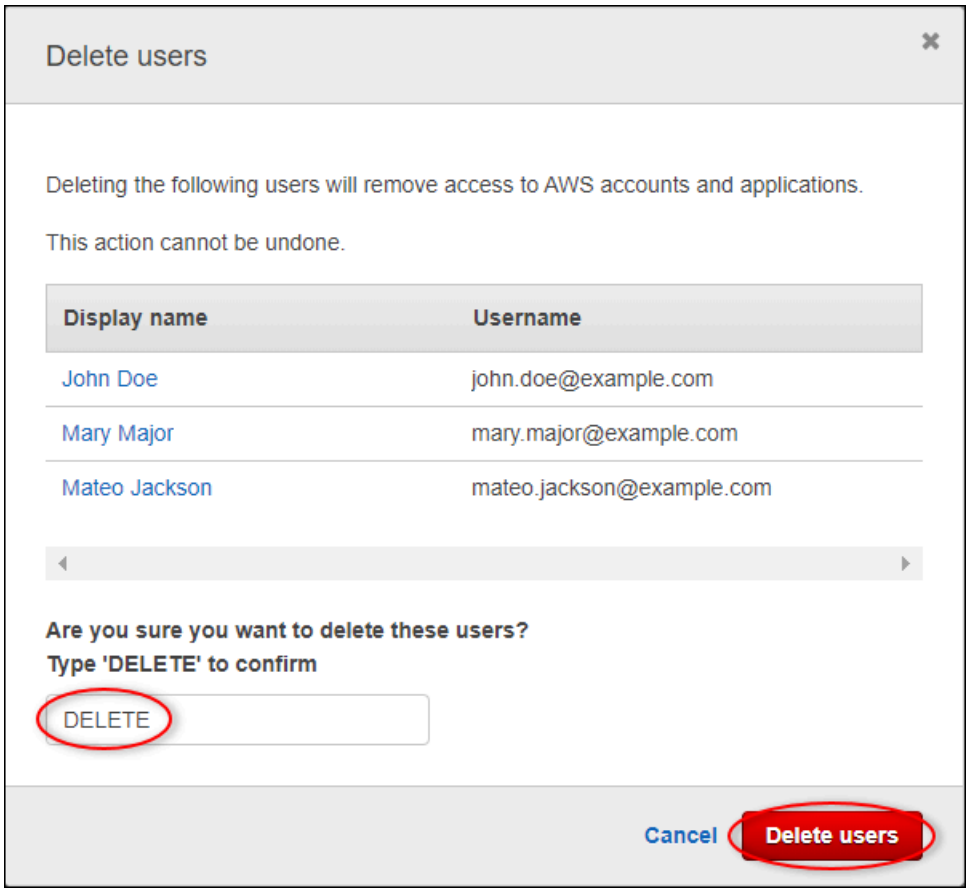
To delete IAM Identity Center users

1. Navigate to the IAM Identity Center console.
2. In the left navigation pane, choose **Users**.

3. Select the check box for each user to delete, and then choose **Delete users**.



4. In the **Delete users** dialog box, enter **DELETE**, and then choose **Delete users**.



Publishing property value updates to Amazon DynamoDB

This tutorial introduces a convenient way to store your data by using Amazon DynamoDB, making it easier to access historical asset data without repeatedly querying the AWS IoT SiteWise API. After you complete this tutorial, you can create custom software that consumes your asset data, such

as a live map of wind speed and direction over an entire wind farm. If you want to monitor and visualize your data without implementing a custom software solution, see Monitoring data with AWS IoT SiteWise Monitor.

In this tutorial, you build on the AWS IoT SiteWise demo that provides a sample set of data for a wind farm. You configure property value updates from the wind farm demo to send data, through AWS IoT Core rules, to a DynamoDB table that you create. When you enable property value updates, AWS IoT SiteWise sends your data to AWS IoT Core in MQTT messages. Then, define AWS IoT Core rules that perform actions, such as the DynamoDB action, depending on the contents of those messages. For more information, see Interacting with other AWS services.

Topics

Prerequisites

To complete this tutorial, you need the following:

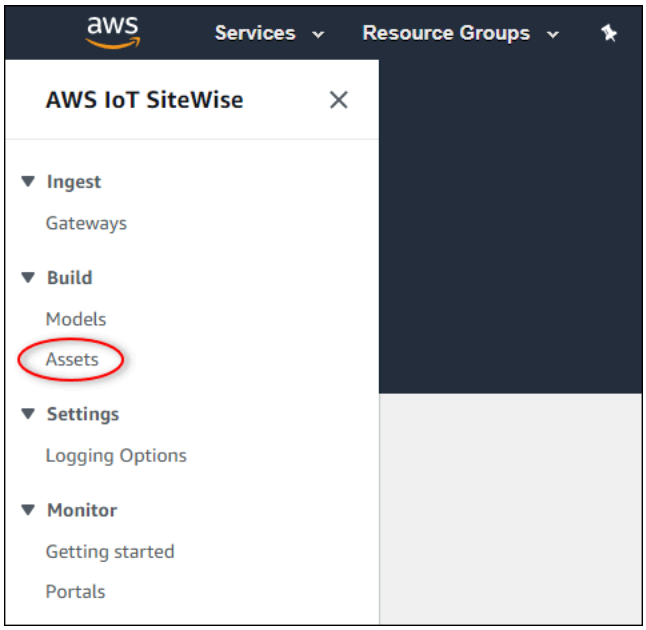
- An AWS account. If you don't have one, see Setting up an AWS account.
- A development computer running Windows, macOS, Linux, or Unix to access the AWS Management Console. For more information, see Getting Started with the AWS Management Console.
- An IAM user with administrator permissions.
- A running AWS IoT SiteWise wind farm demo. When you set up the demo, it defines models and assets in AWS IoT SiteWise and streams data to them to represent a wind farm. For more information, see Using the AWS IoT SiteWise demo.

Step 1: Configure AWS IoT SiteWise to publish property value updates

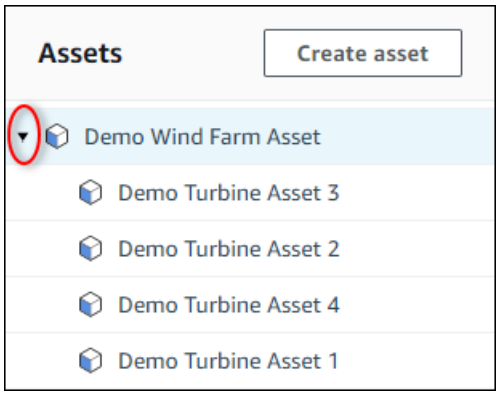
In this procedure, you enable property value notifications on your demo turbine assets' **Wind Speed** properties. After you enable property value notifications, AWS IoT SiteWise publishes each value update in an MQTT message to AWS IoT Core.

To enable property value update notifications on asset properties

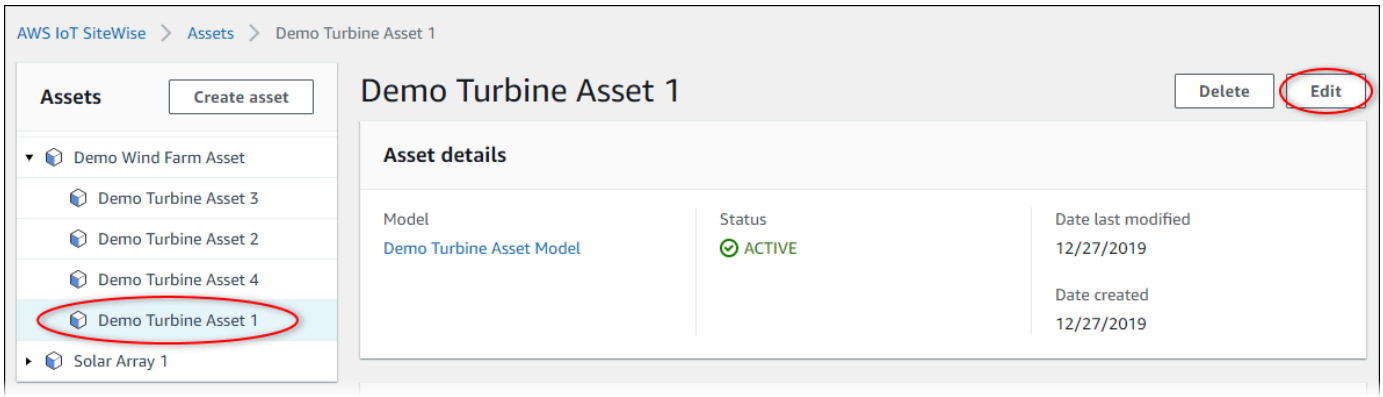
1. Sign in to the AWS IoT SiteWise console.
2. Review the AWS IoT SiteWise endpoints and quotas where AWS IoT SiteWise is supported and switch AWS Regions, if necessary. Switch to a Region where you're running the AWS IoT SiteWise demo.
3. In the left navigation pane, choose **Assets**.



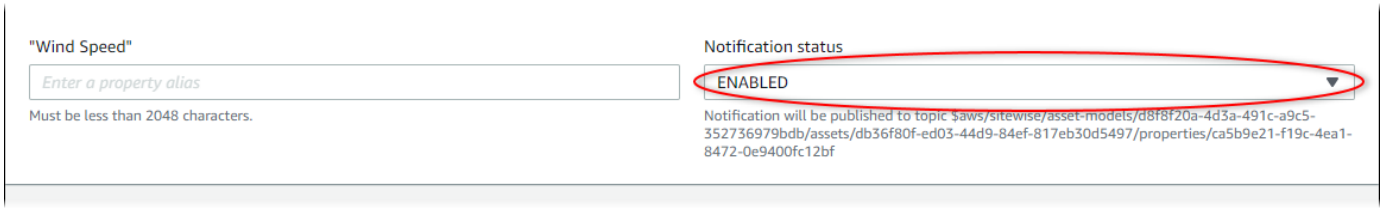
4. Choose the arrow next to **Demo Wind Farm Asset** to expand the wind farm asset's hierarchy.



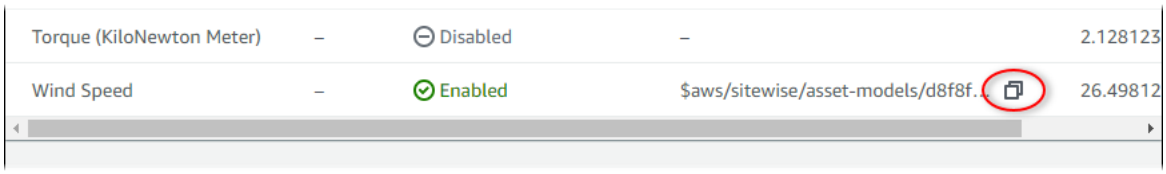
5. Choose a demo turbine and choose **Edit**.



6. Update the **Wind Speed** property's **Notification status** to **ENABLED**.



7. Choose **Save asset** at the bottom of the page.
8. Repeat steps 5 through 7 for each demo turbine asset.
9. Choose a demo turbine (for example, **Demo Turbine Asset 1**).
10. Choose **Measurements**.
11. Choose the copy icon next to the **Wind Speed** property to copy the notification topic to your clipboard. Save the notification topic to use later in this tutorial. You only need to record the notification topic from one turbine.



The notification topic should look like the following example.

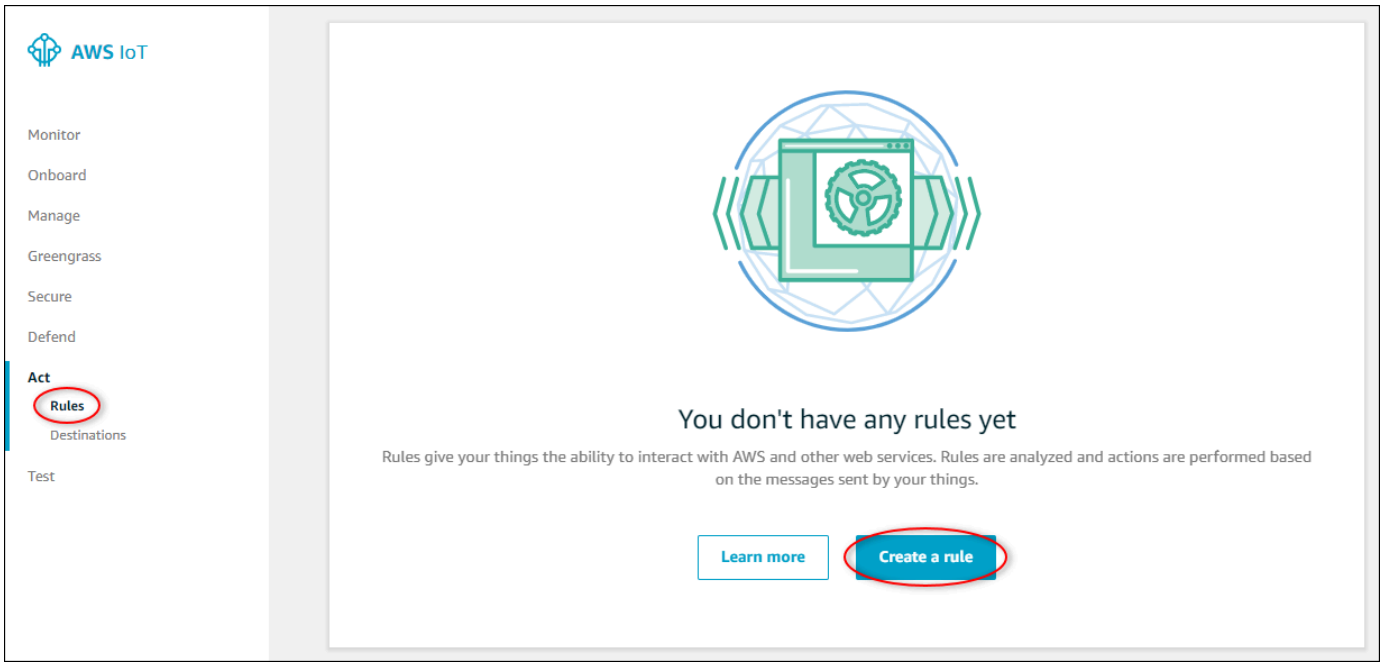
```
$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/
assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE
```

Step 2: Create a rule in AWS IoT Core

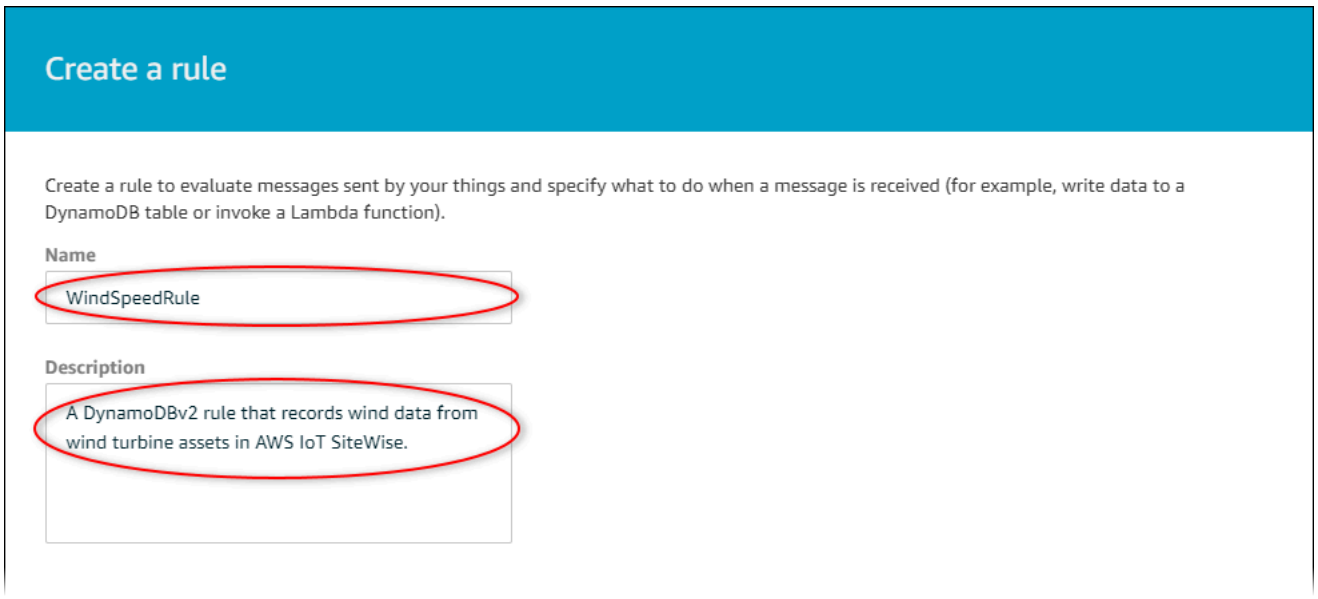
In this procedure, you create a rule in AWS IoT Core that parses the property value notification messages and inserts data into an Amazon DynamoDB table. AWS IoT Core rules parse MQTT messages and perform actions based on the contents and topic of each message. Then, you create a rule with a DynamoDB action to insert data to a DynamoDB table that you create as part of this tutorial.

To create a rule with a DynamoDB action

1. Navigate to the AWS IoT console. If a **Get started** button appears, choose it.
2. In the left navigation pane, choose **Act** and then choose **Rules**.



3. If a **You don't have any rules yet** dialog box appears, choose **Create a rule**. Otherwise, choose **Create**.
4. Enter a name and description for the rule.



5. Find the notification topic that you saved earlier in this tutorial.

```
$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/
assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE
```

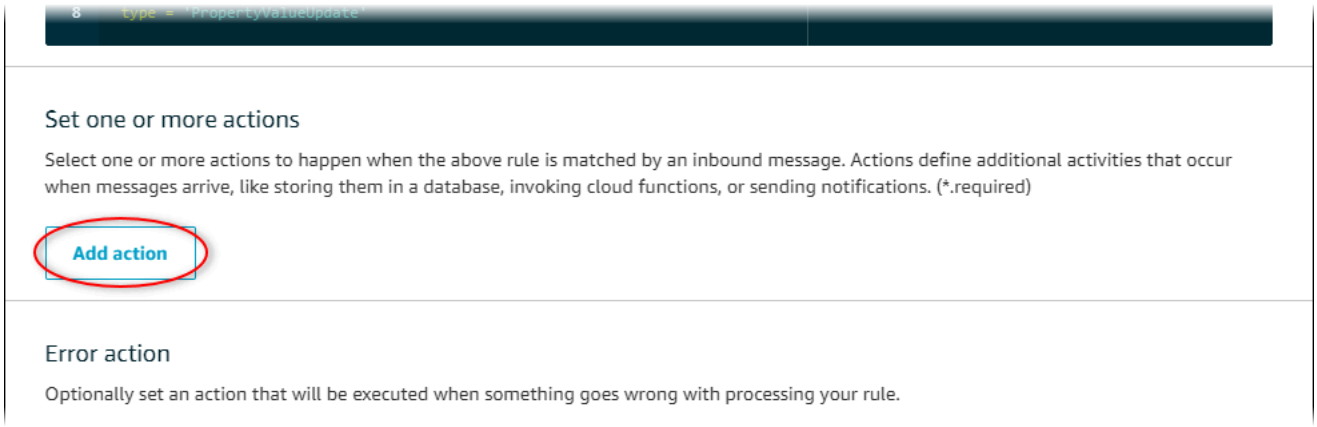
   Replace the asset ID (the ID after `assets/`) in the topic with a +. This selects the wind speed property for all demo wind turbine assets. The + topic filter accepts all nodes from a single level in a topic. Your topic should look like the following example.

```
$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/assets/+/
properties/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE
```

6. Enter the following rule query statement. Replace the topic in the `FROM` section with your notification topic.

```
SELECT
  payload.assetId AS asset,
  (SELECT VALUE (value.doubleValue) FROM payload.values) AS windspeed,
  timestamp() AS timestamp
FROM
  '$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/assets/+/
properties/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE'
WHERE
  type = 'PropertyValueUpdate'
```

7. Under **Set one or more actions**, choose **Add action**.



8. On the **Select an action** page, choose **Split message into multiple columns of a DynamoDB table (DynamoDBv2)**.

9. Choose **Configure action** at the bottom of the page.
10. On the **Configure action** page, choose **Create a new resource**.

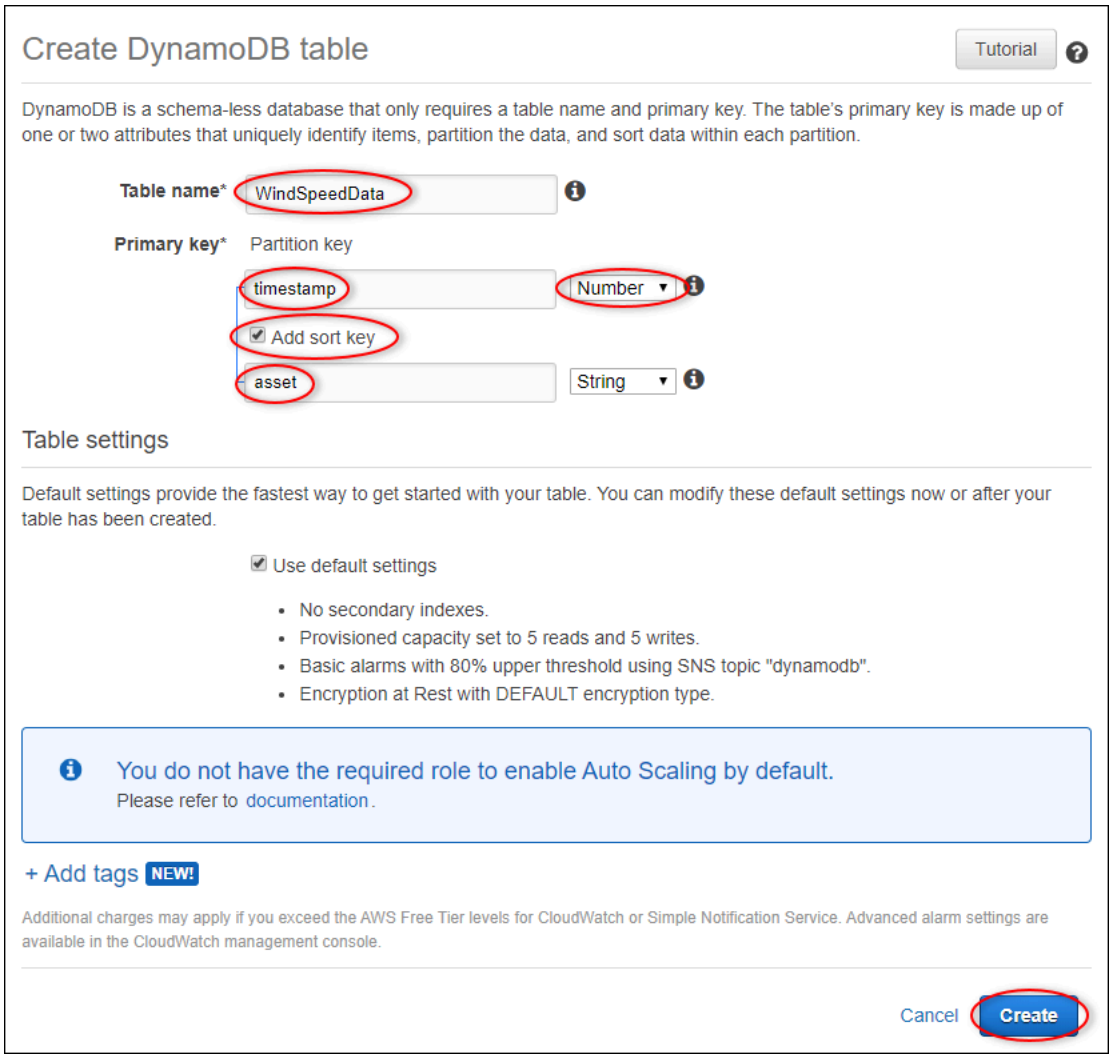
    The DynamoDB console opens in a new tab. Keep the rule action tab open while you complete the following procedures.

Step 3: Create a DynamoDB table

In this procedure, you create an Amazon DynamoDB table to receive wind speed data from the rule action.

To create a DynamoDB table

1. In the DynamoDB console dashboard, choose **Create table**.
2. Enter a name for your table.



3. For **Primary key**, do the following:
   a. Enter **`timestamp`** as the partition key.
   b. Choose the **Number** type.
   c. Select the **Add sort key** check box.
   d. Enter **`asset`** as the sort key, and leave the default sort key type of **String**.
4. Choose **Create**.

   When the **Table is being created** notice disappears, your table is ready.

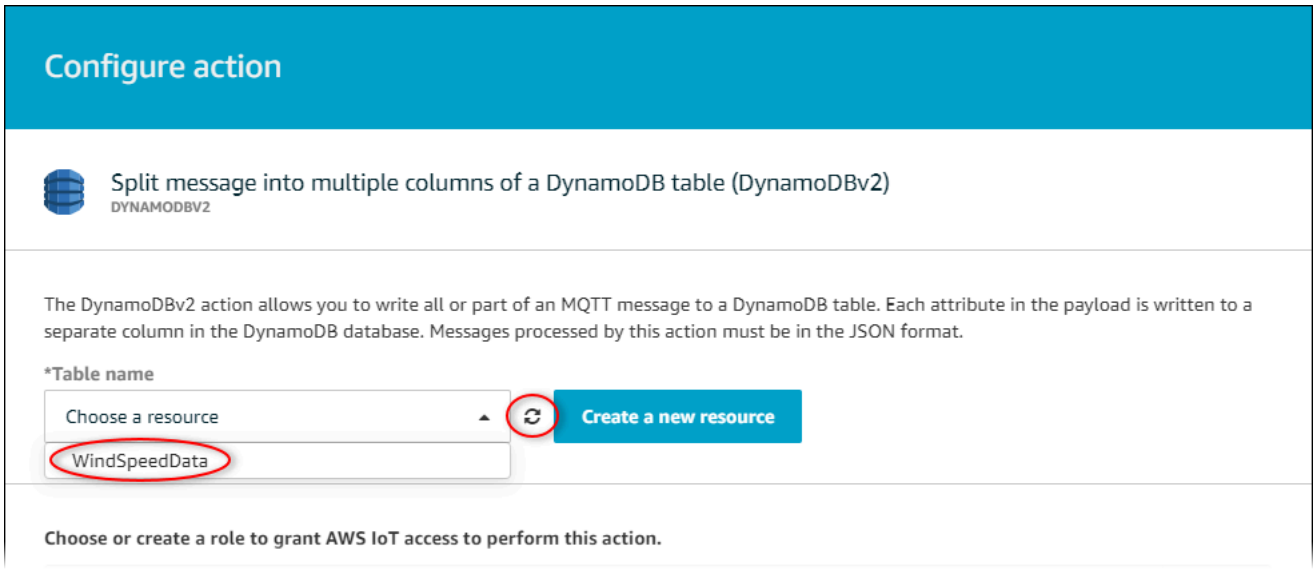
5. Return to the tab with the **Configure action** page. Keep the DynamoDB tab open while you complete the following procedures.

Step 4: Configure the DynamoDB rule action

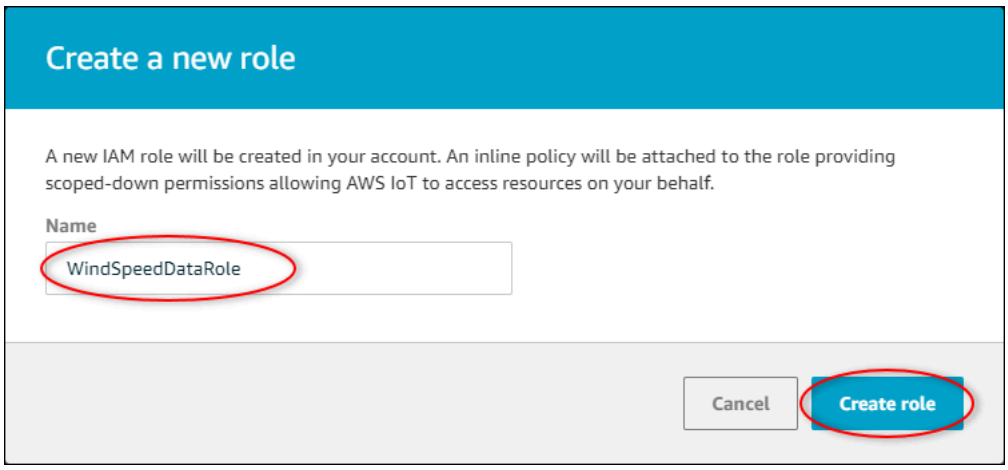
In this procedure, you configure the Amazon DynamoDB rule action to insert data from property value updates to your new DynamoDB table.

To configure the DynamoDB rule action

1. On the **Configure action** page, refresh the **Table name** list, and choose your new DynamoDB table.



2. Choose **Create role** to create an IAM role that grants AWS IoT Core access to perform the rule action.
3. Enter a role name and choose **Create role**.



4. Choose **Add action**.
5. Choose **Create rule** at the bottom of the page to finish creating the rule.

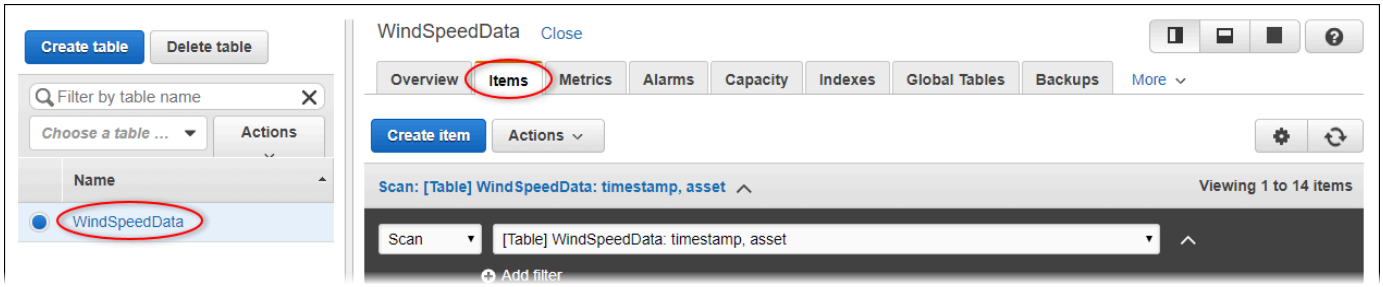
   Your demo asset data should start appearing in your DynamoDB table.

Step 5: Explore data in DynamoDB

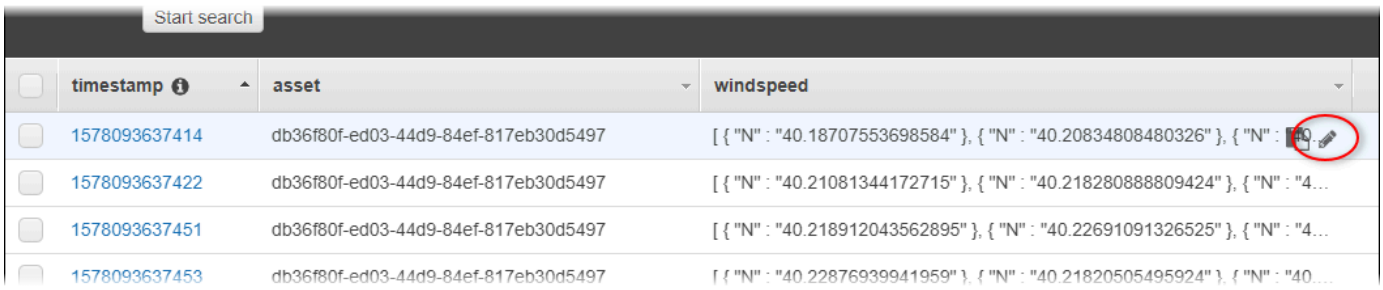
In this procedure, you explore the demo assets' wind speed data in your new Amazon DynamoDB table.

To explore asset data in DynamoDB

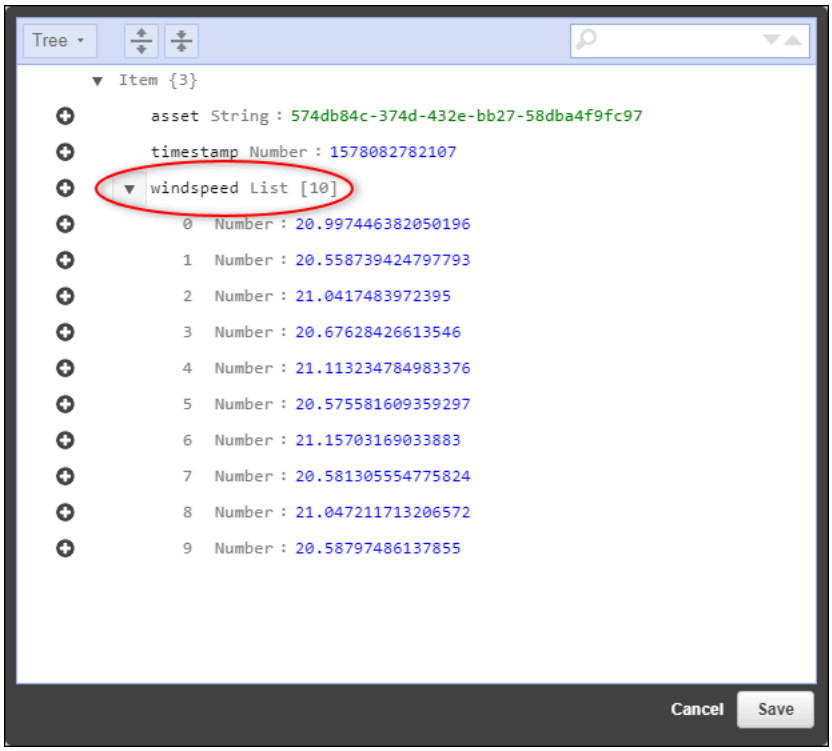
1. Return to the tab with the DynamoDB table open.
2. In the table you created earlier, choose the **Items** tab to view the data in the table. Refresh the page if you don't see rows in the table. If rows don't appear after a few minutes, see Troubleshooting a rule.



3. In a row in the table, choose the edit icon to expand the data.



4. Choose the arrow next to the **windspeed** structure to expand the list of wind speed data points. Each list reflects a batch of wind speed data points sent to AWS IoT SiteWise by the wind farm demo. You might want a different data format if you set up a rule action for your own use. For more information, see Querying asset property notification messages.



Now that you have completed the tutorial, disable or delete the rule and delete your DynamoDB table to avoid incurring additional charges. To clean up your resources, see Step 6: Clean up resources after the tutorial.

Step 6: Clean up resources after the tutorial

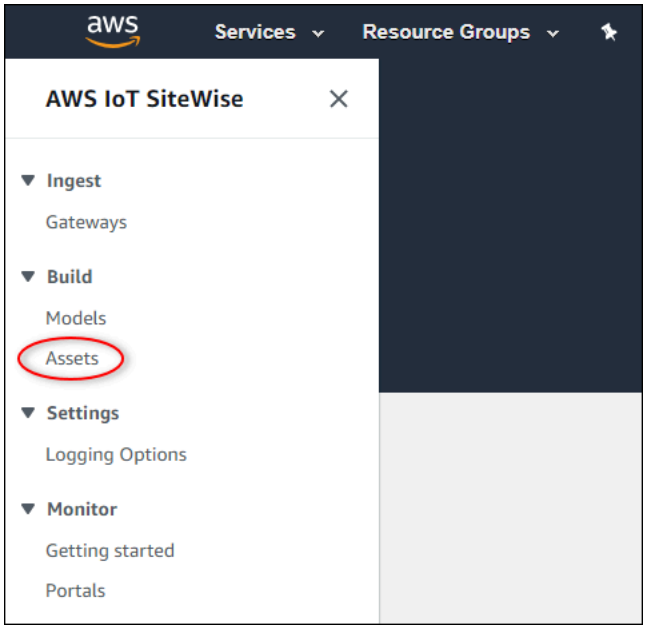
After you complete the tutorial, clean up your resources to avoid incurring additional charges. Your demo wind farm assets are deleted at the end of the duration that you chose when you created

the demo. You can also delete the demo manually. For more information, see Deleting the AWS IoT SiteWise demo.

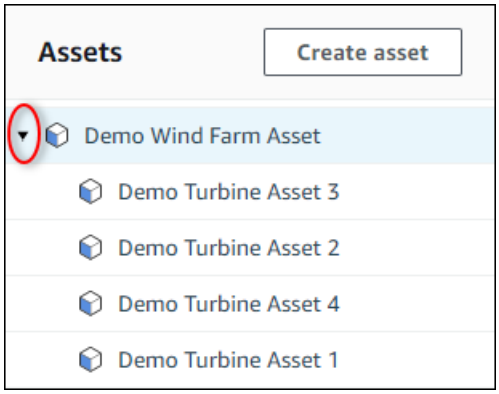
Use the following procedures to disable property value update notifications (if you didn't delete the demo), disable or delete your AWS IoT rule, and delete your DynamoDB table.

To disable property value update notifications on asset properties

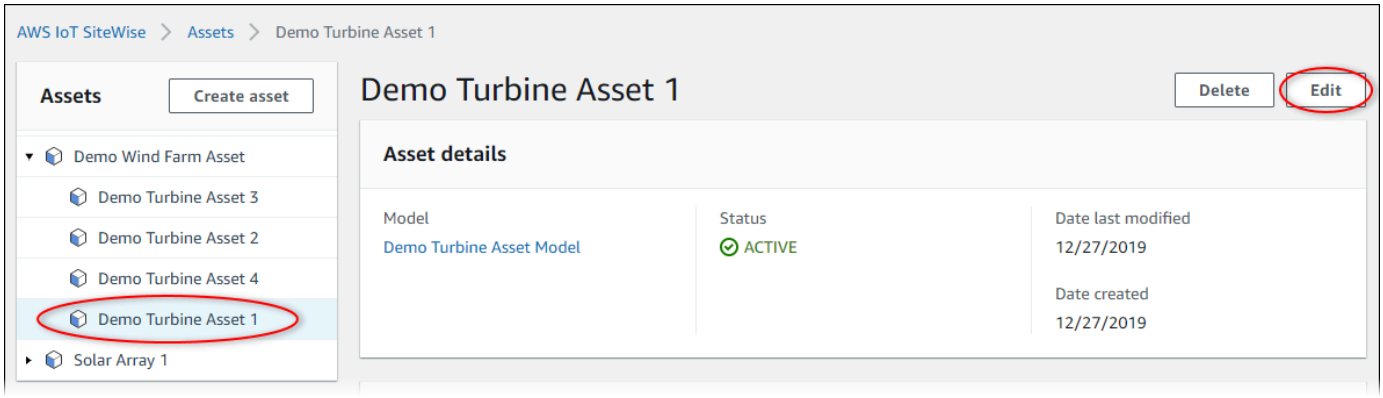
1. Navigate to the AWS IoT SiteWise console.
2. In the left navigation pane, choose **Assets**.



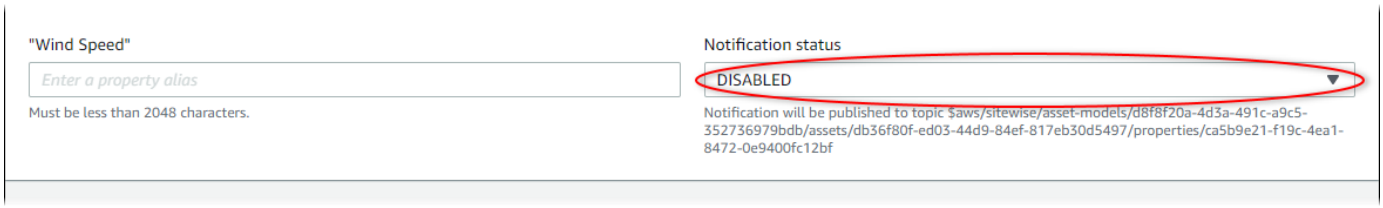
3. Choose the arrow next to **Demo Wind Farm Asset** to expand the wind farm asset's hierarchy.



4. Choose a demo turbine and choose **Edit**.



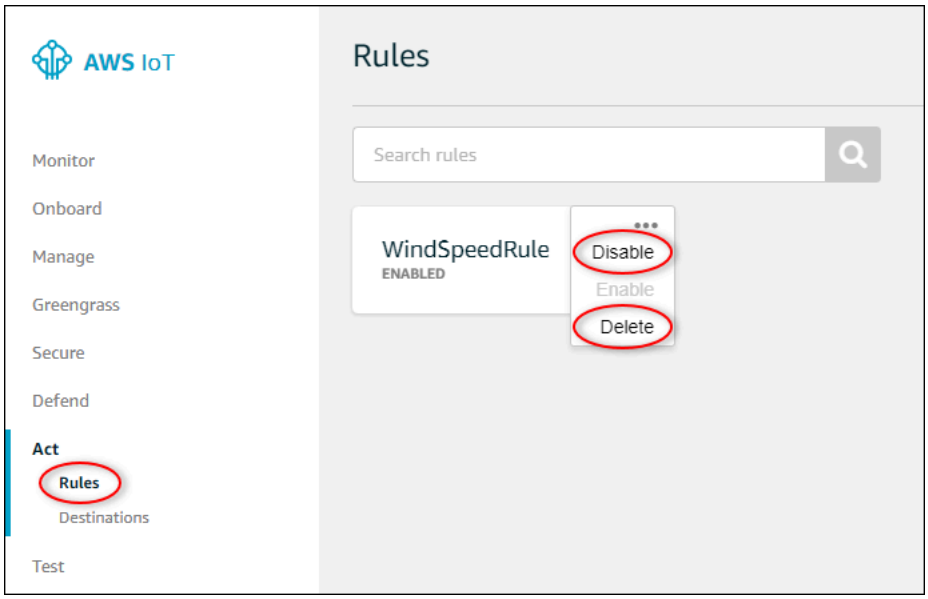
5. Update the **Wind Speed** property's **Notification status** to **DISABLED**.



6. Choose **Save asset** at the bottom of the page.
7. Repeat steps 4 through 6 for each demo turbine asset.

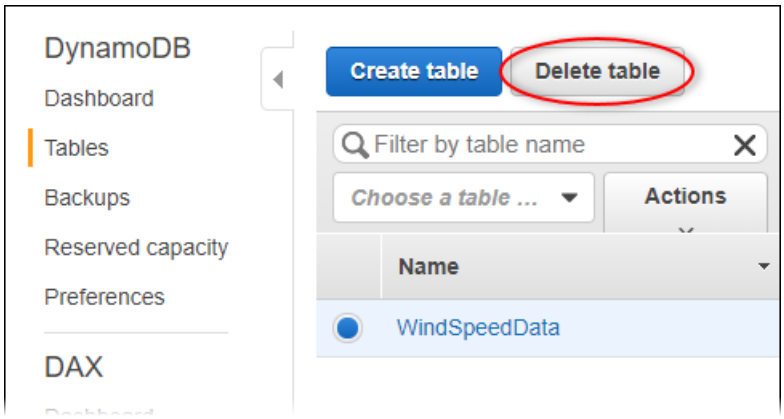
To disable or delete a rule in AWS IoT Core

1. Navigate to the AWS IoT console.
2. In the left navigation pane, choose **Act** and then choose **Rules**.
3. Choose the menu on your rule and choose **Disable** or **Delete**.

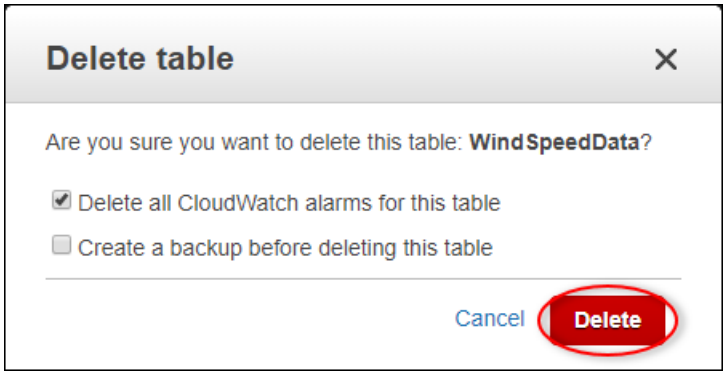


To delete a DynamoDB table

1. Navigate to the DynamoDB console.
2. In the left navigation pane, choose **Tables**.
3. Choose the table you created earlier, **WindSpeedData**.
4. Choose **Delete table**.



5. In the **Delete table** dialog, choose **Delete**.



Ingesting data to AWS IoT SiteWise

AWS IoT SiteWise is designed to efficiently collect and correlate industrial data with corresponding assets, representing various aspects of industrial operations. This documentation focuses on the practical aspects of ingesting data into AWS IoT SiteWise, offering multiple methods tailored to diverse industrial use cases. For instructions to build your virtual industrial operation, see Modeling industrial assets.

You can send industrial data to AWS IoT SiteWise using any of the following options:

- **AWS IoT SiteWise Edge**–Use SiteWise Edge gateway as an intermediary between AWS IoT SiteWise and your data servers. AWS IoT SiteWise provides AWS IoT Greengrass components that you can deploy on any platform that can run AWS IoT Greengrass to set up a SiteWise Edge gateway. This option supports linking with OPC-UA server protocol.
- **AWS IoT SiteWise API**–Use the AWS IoT SiteWise API to upload data from any other source. Use our streaming BatchPutAssetPropertyValue API for ingestion within seconds, or the batch-oriented CreateBulkImportJob API to facilitate cost-effective ingestion in larger batches.
- **AWS IoT Core rules**–Use AWS IoT Core rules to upload data from MQTT messages published by an AWS IoT thing or another AWS service.
- **AWS IoT Events actions**–Use AWS IoT Events actions triggered by specific events in AWS IoT Events. This method is suitable for scenarios where data upload is tied to event occurrences.
- **AWS IoT Greengrass stream manager**–Use AWS IoT Greengrass stream manager to upload data from local data sources using an edge device. This option caters to situations where data originates from on-premises or edge locations.

These methods offer a range of solutions for managing data from different sources. Delve into the details of each option to gain a comprehensive understanding of the data ingestion capabilities AWS IoT SiteWise provides.

Managing data streams

Before you dive into creating asset models and assets in AWS IoT SiteWise, start by setting up your data sources to send information directly from your industrial equipment to the platform. AWS IoT SiteWise is designed to automatically generate data streams that collect your raw data. Each of the data streams is identified by a unique alias, making it easier to keep track of the origin for each piece of data.

For example, consider a wind farm using an AWS IoT SiteWise Edge gateway to send data on air temperature, propeller rotation speed, and power output time-series data from an OPC-UA server to AWS IoT SiteWise. The `server1-windfarm/3/turbine/7/temperature` data stream alias identifies temperature values coming from turbine #7 in wind farm #3. `server1` is the name of the OPC-UA data source. The `server1` prefix is used for all data streams coming from this server, helping to organize data by its source.

After you create the asset models and assets, organize the influx of data by associating each data stream with specific asset properties. This association allows AWS IoT SiteWise to not just collect, but also to process the data according to the structure of your assets. If necessary, you can also remove the link between data streams and asset properties.

Currently, you can only associate data streams with measurements. *Measurements* are a type of asset property that represent devices' raw sensor data streams, such as timestamped temperature values or timestamped rotations per minute (RPM) values.

When these measurements define metrics or transformations, the incoming data triggers specific calculations. It’s important to note that an asset property can only be linked to one data stream at a time.

> **Note**
>
> An asset property can't be associated with multiple data streams at the same time.

AWS IoT SiteWise uses `TimeSeries` for the Amazon Resource Name (ARN) resource to determine your storage charges. For more information, see AWS IoT SiteWise Pricing.

The following sections show you how to use the AWS IoT SiteWise console or API to manage data streams.

Topics

Manage data streams

To begin managing data streams, complete the following.

> **Note**
>
> If you're new to AWS IoT SiteWise after November 24, 2021, you can skip this section. Customers who began using AWS IoT SiteWise before this date need to configure the service settings to allow AWS IoT SiteWise to ingest data without asset models and assets.

- Make sure that your IAM role has the permissions shown in the following example.

  Example IAM user policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PutAssetPropertyValuesAssetPropertyOnly",
            "Effect": "Allow",
            "Action": "iotsitewise:BatchPutAssetPropertyValue",
            "Resource": "arn:aws:iotsitewise:*:*:asset/*"
        },
        {
            "Sid": "PutAssetPropertyValuesPropertyAliasAllowed",
            "Effect": "Allow",
            "Action": "iotsitewise:BatchPutAssetPropertyValue",
            "Resource": "arn:aws:iotsitewise:*:*:time-series/*"
        }
    ]
}
```

> **Important**
>
> Before you ingest data to a data stream, do the following.
>
> - The `time-series` resource must be authorized if you use a property alias to identify the data stream.
> - The `asset` resource must be authorized if you use an asset ID to identify the asset that contains the associated asset property.

For more information about configuring IAM policies, see Managing IAM policies in the *IAM User Guide*.

- Configure data ingestion settings to allow AWS IoT SiteWise to accept data streams that aren't associated with asset properties .

Topics

Configuring data ingestion settings

Console

Configure AWS IoT SiteWise to accept data streams not associated with asset properties by using the AWS IoT SiteWise console.

To configure data ingestion settings (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, under **Settings**, choose **Data ingestion**.
3. On the **Data ingestion** page, choose **Edit**.
4. In the **Disassociated data ingestion** section, choose **Enable data ingestion for data streams not associated with asset properties**.

   > ⚠ **Important**
   >
   > After you configure AWS IoT SiteWise to accept data streams not associated with asset properties, you can't turn off this setting.

5. Choose **Save**.
6. In **Enable disassociated data ingestion**, choose **Update**. The status for **Disassociated data ingestion** becomes **Active**. This process can take a few minutes to complete.

AWS CLI

Configure AWS IoT SiteWise to accept data streams not associated with asset properties by using the PutStorageConfiguration API operation. The following section uses the AWS CLI.

To configure data ingestion settings (AWS CLI)

1. To configure AWS IoT SiteWise to receive data streams not associated with asset properties, run the following command.

   > **Important**
   >
   > After you configure AWS IoT SiteWise to accept data streams not associated with asset properties, you can't turn off this setting.

   ```
   aws iotsitewise put-storage-configuration \
                                   -\-storage-type SITEWISE_DEFAULT_STORAGE \
                                   -\-disassociated-data-storage ENABLED
   ```

   You can configure the `storageType` to `MULTI_LAYER_STORAGE`. For more information, see Managing data storage.

   Example response

   ```
   {
               "storageType": "SITEWISE_DEFAULT_STORAGE",
               "disassociatedDataStorage": "ENABLED",
               "configurationStatus": {
                   "state": "UPDATE_IN_PROGRESS"
               }
           }
   ```

   This process can take a few minutes to complete.

2. To retrieve the storage configuration information, run the following command.

   ```
   aws iotsitewise describe-storage-configuration
   ```

Example response

```
{
            "storageType": "SITEWISE_DEFAULT_STORAGE",
            "disassociatedDataStorage": "ENABLED",
            "configurationStatus": {
                "state": "ACTIVE"
            },
            "lastUpdateDate": "2021-11-16T15:54:14-07:00"
        }
```

Managing data streams

Manage your data streams using the AWS IoT SiteWise console or AWS CLI.

Console

Use the AWS IoT SiteWise console to manage your data streams.

To manage data streams (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Data streams**.
3. (Optional) To add or update tags, select the data stream to edit, then choose **Manage tags**.

   On the **Edit tags** page, choose **Add tag**. In the **Key** field, type the name of the tag to use.

   Choose **Save**.
4. (Optional) In the **Data stream** table, you can filter data streams in the following ways.

   - In the first dropdown menu, select **Alias prefix** or **Asset ID**.
     - **Alias prefix** – The alias prefix of the data stream. You might choose this option if your target data streams have an alias prefix.
     - **Asset ID** – The ID of the asset in which the asset property was created. You might choose this option if your target data streams are associated with an asset property.
   - In the second dropdown menu, select **All data streams**, **Associated data streams**, or **Disassociated data streams**.

- **All data streams** – Data streams that are associated with or not associated with an asset property.
- **Associated data streams** – Data streams that are associated with an asset property.
- **Disassociated data streams** – Data streams that aren't associated with an asset property.

5. Select the data streams that you're managing. AWS IoT SiteWise displays the data streams that you chose in a graph at the bottom of the page. If you select more than 10, the graph will display only the first 10.
6. (Optional) Configure the graph in the following ways.

   a. For **Aggregation function**, select one of the following.

      - **Data point count** – The total number of data points for the given variables over the current time interval.
      - **Average** – The mean of the given variables' values over the current time interval.
      - **Sum** – The sum of the given variables' values over the current time interval.
      - **Minimum** – The minimum of the given variables' values over the current time interval.
      - **Maximum** – The maximum of the given variables' values over the current time interval.

      For more information, see Using aggregation functions in formula expressions.

   b. For **Time ranges**, select one of the following.

      - **Last 1 hour** – The graph displays aggregated data over the last hour.
      - **Last 2 hours** – The graph displays aggregated data over the last two hours.
      - **Last 3 hours** – The graph displays aggregated data over the last three hours.
      - **Last 4 hours** – The graph displays aggregated data over the last four hours.

   c. For **Time interval**, select one of the following.

      - **1 minute** – Aggregates data every minute over the specified time range.
      - **1 hour** – Aggregates data every hour over the specified time range.

7. Choose **Manage data streams**.

8. In the **Update data stream associations** section, in the **Measurement name** column, do one of the following.
   - If the data stream is associated with a measurement, delete the association by choosing the close icon.
   - If the data stream isn't associated with a measurement, choose **Choose measurement**.
9. In the **Choose a measurement** table, navigate to the target asset, and then choose the measurement that you're associating.
10. (Optional) In the **Update asset property aliases** section, enter a unique alias for each measurement.
11. Choose **Update**.

The **Status** column can display one of the following values.

- **Pending** – You're updating the data stream association or asset property alias.
- **Submit** – Your change to the association or asset property alias is saved.
- **Error** – AWS IoT SiteWise couldn't process your request to update the data stream association or the alias for the measurement.
- **Success** – You successfully updated the data stream association or the alias for the measurement.

AWS CLI

Use the following API operations to manage your data streams. The code examples use the AWS CLI.

- AssociateTimeSeriesToAssetProperty – Associates a data stream (time series) with an asset property.
- DisassociateTimeSeriesFromAssetProperty – Disassociates a data stream from an asset property.
- DeleteTimeSeries – Deletes a data stream.
- DescribeTimeSeries – Retrieves information about a data stream.
- ListTimeSeries – Retrieves a paginated list of data streams.

AssociateTimeSeriesToAssetProperty

To associate a data stream with an asset property, run the following command.

> **Important**
>
> The specified asset property must not be currently associated with any data stream.

- Replace *data-stream-alias* with the alias of the data stream that you're associating.
- Replace *asset-ID* with the ID of the asset in which the asset property was created.
- Replace *property-ID* with the ID of the asset property.

```
aws iotsitewise associate-time-series-to-asset-property \
                        --alias data-stream-alias \
                        --assetId asset-ID \
                        --propertyId property-ID
```

DisassociateTimeSeriesFromAssetProperty

To disassociate a data stream from an asset property, run the following command.

- Replace *data-stream-alias* with the alias of the data stream that you're disassociating.
- Replace *asset-ID* with the ID of the asset in which the asset property was created.
- Replace *property-ID* with the ID of the asset property.

```
aws iotsitewise disassociate-time-series-from-asset-property \
                        --alias data-stream-alias \
                        --assetId asset-ID \
                        --propertyId property-ID
```

DeleteTimeSeries

To delete a data stream, run the following command.

Replace *data-stream-alias* with the alias of the data stream that you're deleting.

```
aws iotsitewise delete-time-series --alias data-stream-alias
```

To identify a data stream, do one of the following:

- If the data stream isn't associated with an asset property, specify the `alias` of the data stream.
- If the data stream is associated with an asset property, specify one of the following:
  - The `alias` of the data stream.
  - The `assetId` and `propertyId` that identifies the asset property.

DescribeTimeSeries

Use the `DescribeTimeSeries` API operation to verify if you successfully associated or disassociated a data stream.

To retrieve information about a data stream, run the following command.

```
aws iotsitewise describe-time-series --alias data-stream-alias
```

To identify a data stream, do one of the following:

- If the data stream isn't associated with an asset property, specify the `alias` of the data stream.
- If the data stream is associated with an asset property, specify one of the following:
  - The `alias` of the data stream.
  - The `assetId` and `propertyId` that identifies the asset property.

ListTimeSeries

Use the `ListTimeSeries` API operation to verify if you successfully deleted a data stream.

To retrieve a paginated list of data streams, run the following command.

```
aws iotsitewise list-time-series
```

Ingesting data using the AWS IoT SiteWise API

Use the AWS IoT SiteWise API to send timestamped industrial data to your assets' attribute and measurement properties. The API accepts a payload that contains timestamp-quality-value (TQV) structures.

Use the BatchPutAssetPropertyValue operation to upload your data. With this operation, you can upload multiple data entries at a time to collect data from several devices and send it all in a single request.

Important

The BatchPutAssetPropertyValue operation is subject to the following quotas:

- Up to 10 entries per request.
- Up to 10 property values (TQV data points) per entry.
- AWS IoT SiteWise rejects any data with a timestamp dated to more than 7 days in the past or more than 10 minutes in the future.

For more information about these quotas, see BatchPutAssetPropertyValue in the *AWS IoT SiteWise API Reference*.

To identify an asset property, specify one of the following:

- The `assetId` and `propertyId` of the asset property that data is sent to.
- The `propertyAlias`, which is a data stream alias (for example, `/company/windfarm/3/turbine/7/temperature`). To use this option, you must first set your asset property's alias. To set property aliases, see Mapping industrial data streams to asset properties.

The following example demonstrates how to send a wind turbine's temperature and rotations per minute (RPM) readings from a payload stored in a JSON file.

```
aws iotsitewise batch-put-asset-property-value --cli-input-json file://batch-put-payload.json
```

The example payload in `batch-put-payload.json` contains the following content.

```
{
  "entries": [
    {
      "entryId": "unique entry ID",
      "propertyAlias": "/company/windfarm/3/turbine/7/temperature",
      "propertyValues": [
```

```
        {
          "value": {
            "integerValue": 38
          },
          "timestamp": {
            "timeInSeconds": 1575691200
          }
        }
      ]
    },
    {
      "entryId": "unique entry ID",
      "propertyAlias": "/company/windfarm/3/turbine/7/rpm",
      "propertyValues": [
        {
          "value": {
            "doubleValue": 15.09
          },
          "timestamp": {
            "timeInSeconds": 1575691200
          },
          "quality": "GOOD"
        }
      ]
    }
  ]
}
```

Each entry in the payload contains an `entryId` that you can define as any unique string. If any request entries fail, each error will contain the `entryId` of the corresponding request so that you know which requests to retry.

Each structure in the list of `propertyValues` is a timestamp-quality-value (TQV) structure that contains a `value`, a `timestamp`, and optionally a `quality`.

- `value` – A structure that contains one of the following fields, depending on the type of the property being set:
  - `booleanValue`
  - `doubleValue`
  - `integerValue`
  - `stringValue`

- `timestamp` – A structure that contains the current Unix epoch time in seconds, `timeInSeconds`. You can also set the `offsetInNanos` key in the `timestamp` structure if you have temporally precise data. AWS IoT SiteWise rejects any data points with timestamps older than 7 days in the past or newer than 10 minutes in the future.
- `quality` – (Optional) One of the following quality strings:
  - `GOOD` – (Default) The data isn't affected by any issues.
  - `BAD` – The data is affected by an issue such as sensor failure.
  - `UNCERTAIN` – The data is affected by an issue such as sensor inaccuracy.

  For more information about how AWS IoT SiteWise handles data quality in computations, see Data quality in formula expressions.

Ingesting data using AWS IoT Core rules

Send data to AWS IoT SiteWise from AWS IoT things and other AWS services by using rules in AWS IoT Core. Rules transform MQTT messages and perform actions to interact with AWS services. The AWS IoT SiteWise rule action forwards messages data to the BatchPutAssetPropertyValue operation from the AWS IoT SiteWise API. For more information, see Rules and AWS IoT SiteWise action in the *AWS IoT Developer Guide*.

To follow a tutorial that walks through the steps required to set up a rule that ingests data through device shadows, see Ingesting data from AWS IoT things.

You can also send data from AWS IoT SiteWise to other AWS services. For more information, see Interacting with other AWS services.

Topics

- Granting AWS IoT the required access
- Configuring the AWS IoT SiteWise rule action
- Reducing costs with basic ingest

Granting AWS IoT the required access

You use IAM roles to control the AWS resources to which each rule has access. Before you create a rule, you must create an IAM role with a policy that allows the rule to perform actions on the required AWS resource. AWS IoT assumes this role when running a rule.

If you create the rule action in the AWS IoT console, you can choose a root asset to create a role that has access to a selected asset hierarchy. For more information about how to manually define a role for a rule, see Granting AWS IoT the required access and Pass role permissions in the *AWS IoT Developer Guide*.

For the AWS IoT SiteWise rule action, you must define a role that allows `iotsitewise:BatchPutAssetPropertyValue` access to the asset properties to which the rule sends data. To improve security, you can specify an AWS IoT SiteWise asset hierarchy path in the `Condition` property.

The following example trust policy allows access to a specific asset and its children.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iotsitewise:BatchPutAssetPropertyValue",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iotsitewise:assetHierarchyPath": [
            "/root node asset ID",
            "/root node asset ID/*"
          ]
        }
      }
    }
  ]
}
```

Remove the `Condition` from the policy to allow access to all of your assets. The following example trust policy allows access to all of your assets in the current Region.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iotsitewise:BatchPutAssetPropertyValue",
      "Resource": "*"
```

```
    }
  ]
}
```

Configuring the AWS IoT SiteWise rule action

The AWS IoT SiteWise rule action sends data from the MQTT message that initiated the rule to asset properties in AWS IoT SiteWise. You can upload multiple data entries to different asset properties at the same time,to send updates for all sensors of a device in one message. You can also upload multiple data points at once for each data entry.

> **Note**
>
> When you send data to AWS IoT SiteWise with the rule action, your data must meet all of the requirements of the `BatchPutAssetPropertyValue` operation. For example, your data can't have a timestamp earlier than 7 days from current Unix epoch time. For more information, see Ingesting data with the AWS IoT SiteWise API.

For each data entry in the rule action, you identify an asset property and specify the timestamp, quality, and value of each data point for that asset property. The rule action expects strings for all parameters.

To identify an asset property in an entry, specify one of the following:

- The **Asset ID** (`assetId`) and **Property ID** (`propertyId`) of the asset property that you're sending data to. You can find the Asset ID and Property ID using the AWS IoT SiteWise console. If you know the Asset ID, you can use the AWS CLI to call DescribeAsset to find the Property ID.
- The **Property alias** (`propertyAlias`), which is a data stream alias (for example, `/company/windfarm/3/turbine/7/temperature`). To use this option, you must first set your asset property's alias. To learn how to set property aliases, see Mapping industrial data streams to asset properties.

For the timestamp in each entry, use the timestamp reported by your equipment or the timestamp provided by AWS IoT Core. The timestamp has two parameters:

- **Time in seconds** (`timeInSeconds`) – The Unix epoch time, in seconds, at which the sensor or equipment reported the data.

- **Offset in nanos** (`offsetInNanos`) – (Optional) The nanosecond offset from the time in seconds.

> **Important**
>
> If your timestamp is a string, has a decimal portion, or isn't in seconds, AWS IoT SiteWise rejects the request. You must convert the timestamp to seconds and nanosecond offset. Use features of the AWS IoT rules engine to convert the timestamp. For more information, see the following:
>
> - Getting timestamps for devices that don't report accurate time
> - Converting timestamps that are in string format

You can use substitution templates for several parameters in the action to perform calculations, invoke functions, and pull values from the message payload. For more information, see Substitution templates in the *AWS IoT Developer Guide*.

> **Note**
>
> Because an expression in a substitution template is evaluated separately from the SELECT statement, you can't use a substitution template to reference an alias created using an AS clause. You can reference only information present in the original payload, in addition to supported functions and operators.

Topics

Getting timestamps for devices that don't report accurate time

If your sensor or equipment doesn't report accurate time data, get the current Unix epoch time from the AWS IoT rules engine with timestamp(). This function outputs time in milliseconds, so you

must convert the value to time in seconds and offset in nanoseconds. To do so, use the following conversions:

- For **Time in seconds** (`timeInSeconds`), use **`${floor(timestamp() / 1E3)}`** to convert the time from milliseconds to seconds.
- For **Offset in nanos** (`offsetInNanos`), use **`${(timestamp() % 1E3) * 1E6}`** to calculate the nanosecond offset of the timestamp.

Converting timestamps that are in string format

If your sensor or equipment reports time data in string format (for example, `2020-03-03T14:57:14.699Z`), use time_to_epoch(String, String). This function inputs the timestamp and format pattern as parameters and outputs time in milliseconds. Then, you must convert the time to time in seconds and offset in nanoseconds. To do so, use the following conversions:

- For **Time in seconds** (`timeInSeconds`), use **`${floor(time_to_epoch("2020-03-03T14:57:14.699Z", "yyyy-MM-dd'T'HH:mm:ss'Z'") / 1E3)}`** to convert the timestamp string to milliseconds, and then to seconds.
- For **Offset in nanos** (`offsetInNanos`), use **`${(time_to_epoch("2020-03-03T14:57:14.699Z", "yyyy-MM-dd'T'HH:mm:ss'Z'") % 1E3) * 1E6}`** to calculate the nanosecond offset of the timestamp string.

> **Note**
>
> The `time_to_epoch` function supports up to millisecond-precision timestamp strings. To convert strings with microsecond or nanosecond precision, configure an AWS Lambda function that your rule calls to convert the timestamp into numerical values. For more information, see Converting nanosecond-precision timestamp strings.

Converting nanosecond-precision timestamp strings

If your device sends timestamp information in string format with nanosecond precision (for example, `2020-03-03T14:57:14.699728491Z`), use the following procedure to configure your rule action. You can create an AWS Lambda function that converts the timestamp from a

string into **Time in seconds** (`timeInSeconds`) and **Offset in nanos** (`offsetInNanos`). Then, use aws_lambda(functionArn, inputJson) in your rule action parameters to invoke that Lambda function and use the output in your rule.

> **Note**
>
> This section contains advanced instructions that assume that you're familiar with how to create the following resources:
>
> - Lambda functions. For more information, see Create a Lambda function with the console or Using Lambda with the AWS CLI in the *AWS Lambda Developer Guide*.
> - AWS IoT rules with the AWS IoT SiteWise rule action. For more information, see Ingesting data using AWS IoT Core rules.

To create an AWS IoT SiteWise rule action that parses timestamp strings

1. Create a Lambda function with the following properties:

   - **Function name** – Use a descriptive function name (for example, **`ConvertNanosecondTimestampFromString`**).
   - **Runtime** – Use a Python 3 runtime, such as **Python 3.11** (`python3.11`).
   - **Permissions** – Create a role with basic Lambda permissions (**AWSLambdaBasicExecutionRole**).
   - **Layers** – Add the **AWSSDKPandas-Python311** layer for the Lambda function to use `numpy`.
   - **Function code** – Use the following function code, which consumes a string argument named `timestamp` and outputs `timeInSeconds` and `offsetInNanos` values for that timestamp.

```
import json
import math
import numpy

# Converts a timestamp string into timeInSeconds and offsetInNanos in Unix epoch
 time.
# The input timestamp string can have up to nanosecond precision.
def lambda_handler(event, context):
    timestamp_str = event['timestamp']
    # Parse the timestamp string as nanoseconds since Unix epoch.
    nanoseconds = numpy.datetime64(timestamp_str, 'ns').item()
```

```
    time_in_seconds = math.floor(nanoseconds / 1E9)
    # Slice to avoid precision issues.
    offset_in_nanos = int(str(nanoseconds)[-9:])
    return {
        'timeInSeconds': time_in_seconds,
        'offsetInNanos': offset_in_nanos
    }
```

This Lambda function inputs timestamp strings in ISO 8601 format using datetime64 from NumPy.

> **Note**
>
> If your timestamp strings aren't in ISO 8601 format, you can implement a solution with pandas that defines the timestamp format. For more information, see pandas.to_datetime.

2. When you configure the AWS IoT SiteWise action for your rule, use the following substitution templates for **Time in seconds** (`timeInSeconds`) and **Offset in nanos** (`offsetInNanos`). These substitution templates assume that your message payload contains the timestamp string in `timestamp`. The `aws_lambda` function consumes a JSON structure for its second parameter, so you can modify the below substitution templates if needed.

   - For **Time in seconds** (`timeInSeconds`), use the following substitution template.

   ```
   ${aws_lambda('arn:aws:lambda:region:account-
   id:function:ConvertNanosecondTimestampFromString', {'timestamp':
    timestamp}).timeInSeconds}
   ```

   - For **Offset in nanos** (`offsetInNanos`), use the following substitution template.

   ```
   ${aws_lambda('arn:aws:lambda:region:account-
   id:function:ConvertNanosecondTimestampFromString', {'timestamp':
    timestamp}).offsetInNanos}
   ```

   For each parameter, replace *region* and *account-id* with your Region and AWS account ID. If you used a different name for your Lambda function, change that as well.

3. Grant AWS IoT permissions to invoke your function with the `lambda:InvokeFunction` permission. For more information, see aws_lambda(functionArn, inputJson).

4. Test your rule (for example, use the AWS IoT MQTT test client) and verify that AWS IoT SiteWise receives the data that you send.

   If your rule doesn't work as expected, see Troubleshooting an AWS IoT SiteWise rule action.

> **Note**
>
> This solution invokes the Lambda function twice for each timestamp string. You can create another rule to reduce the number of Lambda function invocations if your rule handles multiple data points that have the same timestamp in each payload.
> To do so, create a rule with a republish action that invokes the Lambda and publishes the original payload with the timestamp string converted to `timeInSeconds` and `offsetInNanos`. Then, create a rule with an AWS IoT SiteWise rule action to consume the converted payload. With this approach, you reduce the number of times that the rule invokes the Lambda but increase the number of AWS IoT rule actions run. Consider the pricing of each service if you apply this solution to your use case.

Example rule configurations

This section contains example rule configurations to create a rule with an AWS IoT SiteWise action.

Example Example rule action that uses property aliases as message topics

The following example creates a rule with an AWS IoT SiteWise action that uses the topic (through topic()) as the property alias to identify asset properties. Use this example to define one rule for ingesting double-type data to all wind turbines in all wind farms. This example requires that you define property aliases on all turbine assets' properties. You would need to define a second, similar rule to ingest integer-type data.

```
aws iot create-topic-rule \
  --rule-name SiteWiseWindFarmRule \
  --topic-rule-payload file://sitewise-rule-payload.json
```

The example payload in `sitewise-rule-payload.json` contains the following content.

```
{
  "sql": "SELECT * FROM '/company/windfarm/+/turbine/+/+' WHERE type = 'double'",
```

```
  "description": "Sends data to the wind turbine asset property with the same alias as
 the topic",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
    {
      "iotSiteWise": {
        "putAssetPropertyValueEntries": [
          {
            "propertyAlias": "${topic()}",
            "propertyValues": [
              {
                "timestamp": {
                  "timeInSeconds": "${timeInSeconds}"
                },
                "value": {
                  "doubleValue": "${value}"
                }
              }
            ]
          }
        ],
        "roleArn": "arn:aws:iam::account-id:role/MySiteWiseActionRole"
      }
    }
  ]
}
```

With this rule action, send the following message to a wind turbine property alias (for example, `/company/windfarm/3/turbine/7/temperature`) as a topic to ingest data.

```
{
  "type": "double",
  "value": "38.3",
  "timeInSeconds": "1581368533"
}
```

Example Example rule action that uses timestamp() to determine time

The following example creates a rule with an AWS IoT SiteWise action that identifies an asset property by IDs and uses timestamp() to determine the current time.

```
aws iot create-topic-rule \
```

```
  --rule-name SiteWiseAssetPropertyRule \
  --topic-rule-payload file://sitewise-rule-payload.json
```

The example payload in `sitewise-rule-payload.json` contains the following content.

```
{
  "sql": "SELECT * FROM 'my/asset/property/topic'",
  "description": "Sends device data to an asset property",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
    {
      "iotSiteWise": {
        "putAssetPropertyValueEntries": [
          {
            "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
            "propertyId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
            "propertyValues": [
              {
                "timestamp": {
                  "timeInSeconds": "${floor(timestamp() / 1E3)}",
                  "offsetInNanos": "${(timestamp() % 1E3) * 1E6}"
                },
                "value": {
                  "doubleValue": "${value}"
                }
              }
            ]
          }
        ],
        "roleArn": "arn:aws:iam::account-id:role/MySiteWiseActionRole"
      }
    }
  ]
}
```

With this rule action, send the following message to the `my/asset/property/topic` to ingest data.

```
{
  "type": "double",
  "value": "38.3"
```

```
}
```

Troubleshooting the rule action

To troubleshoot your AWS IoT SiteWise rule action in AWS IoT Core, configure CloudWatch Logs or configure a republish error action for your rule. For more information, see Troubleshooting an AWS IoT SiteWise rule action.

Reducing costs with basic ingest

AWS IoT Core provides a feature called Basic Ingest that you can use to send data through AWS IoT Core without incurring AWS IoT messaging costs. Basic Ingest optimizes data flow for high volume data ingestion workloads by removing the publish/subscribe message broker from the ingestion path. You can use Basic Ingest if you know which rules your messages should be routed to.

To use Basic Ingest, you send messages directly to a specific rule using a special topic, `$aws/rules/rule-name`. For example, to send a message to a rule named `SiteWiseWindFarmRule`, you send a message to the topic `$aws/rules/SiteWiseWindFarmRule`.

If your rule action uses substitution templates that contain topic(Decimal), you can pass the original topic at the end of the Basic Ingest special topic, such as `$aws/rules/rule-name/original-topic`. For example, to use Basic Ingest with the wind farm property alias example from the previous section, you can send messages to the following topic.

```
$aws/rules/SiteWiseWindFarmRule//company/windfarm/3/turbine/7/temperature
```

> **Note**
>
> The above example includes a second slash (`//`) because AWS IoT removes the Basic Ingest prefix (`$aws/rules/rule-name/`) from the topic that's visible to the rule action. In this example, the rule receives the topic `/company/windfarm/3/turbine/7/temperature`.

For more information, see Reducing messaging costs with basic ingest in the *AWS IoT Developer Guide*.

Ingesting data from AWS IoT Events

With AWS IoT Events, you can build complex event monitoring applications for your IoT fleet in the AWS Cloud. Use the IoT SiteWise action in AWS IoT Events to send data to asset properties in AWS IoT SiteWise when an event occurs.

AWS IoT Events is designed to streamline the development of event monitoring applications for IoT devices and systems within the AWS Cloud. Using AWS IoT Events, you can:

- Detect and respond to changes, anomalies, or specific conditions across your IoT fleet.
- Enhance your operational efficiency and enable proactive management of your IoT ecosystem.

By integrating with AWS IoT SiteWise through the AWS IoT SiteWise action, AWS IoT Events extends its capabilities, allowing you to automatically update asset properties in AWS IoT SiteWise in response to specific events. This interaction can simplify data ingestion and management. It can also empower you with actionable insights.

For more information, see the following topics in the *AWS IoT Events Developer Guide*:

- What is AWS IoT Events?
- AWS IoT Events actions
- IoT SiteWise action

Using AWS IoT Greengrass stream manager

AWS IoT Greengrass stream manager is an integration feature that facilitates the transfer of data streams from local sources to the AWS Cloud. It acts as an intermediary layer that manages data flows, enabling devices operating at the edge to gather and store data before it is sent to AWS IoT SiteWise, for further analysis and processing.

Add a data destination by configuring a local source on the AWS IoT SiteWise console. You can also use stream manager in your custom AWS IoT Greengrass solution to ingest data to AWS IoT SiteWise.

> **Note**
>
> To ingest data from OPC-UA sources, configure an AWS IoT SiteWise Edge gateway that runs on AWS IoT Greengrass. For more information, see Using SiteWise Edge gateways.

For more information about how to **configure a destination** for local source data, see Configuring data sources.

For more information about how to **ingest data using stream manager** in a custom AWS IoT Greengrass solution, see the following topics in the *AWS IoT Greengrass Version 2 Developer Guide*:

- What is AWS IoT Greengrass?
- Manage data streams on the AWS IoT Greengrass core
- Exporting data to AWS IoT SiteWise asset properties

Ingesting data using the CreateBulkImportJob API

Use the `CreateBulkImportJob` API to import large amounts of data from Amazon S3. Your data must be saved in the CSV format in Amazon S3. Data files can have the following columns.

> **Note**
>
> To identify an asset property, specify one of the following.
>
> - The `ASSET_ID` and `PROPERTY_ID` of the asset property that you you're sending data to.
> - The `ALIAS`, which is a data stream alias (for example, `/company/windfarm/3/turbine/7/temperature`). To use this option, you must first set your asset property's alias. To learn how to set property aliases, see the section called “Mapping industrial data streams to asset properties”.

- `ALIAS` – The alias that identifies the property, such as an OPC-UA server data stream path (for example, `/company/windfarm/3/turbine/7/temperature`). For more information, see Mapping industrial data streams to asset properties.
- `ASSET_ID` – The ID of the asset.

- PROPERTY_ID – The ID of the asset property.
- DATA_TYPE – The property's data type can be one of the following.
  - STRING – A string with up to 1024 bytes.
  - INTEGER – A signed 32-bit integer with range [-2,147,483,648, 2,147,483,647].
  - DOUBLE – A floating point number with range [-10^100, 10^100] and IEEE 754 double precision.
  - BOOLEAN – `true` or `false`.
- TIMESTAMP_SECONDS – The timestamp of the data point, in Unix epoch time.
- TIMESTAMP_NANO_OFFSET – The nanosecond offset coverted from TIMESTAMP_SECONDS.
- QUALITY – (Optional) The quality of the asset property value. The value can be one of the following.
  - GOOD – (Default) The data isn't affected by any issues.
  - BAD – The data is affected by an issue such as sensor failure.
  - UNCERTAIN – The data is affected by an issue such as sensor inaccuracy.

  For more information about how AWS IoT SiteWise handles data quality in computations, see Data quality in formula expressions.
- VALUE – The value of the asset property.

Example data file(s) in the .csv format

```
asset_id,property_id,DOUBLE,1635201373,0,GOOD,1.0
asset_id,property_id,DOUBLE,1635201374,0,GOOD,2.0
asset_id,property_id,DOUBLE,1635201375,0,GOOD,3.0
```

```
unmodeled_alias1,DOUBLE,1635201373,0,GOOD,1.0
unmodeled_alias1,DOUBLE,1635201374,0,GOOD,2.0
unmodeled_alias1,DOUBLE,1635201375,0,GOOD,3.0
unmodeled_alias1,DOUBLE,1635201376,0,GOOD,4.0
unmodeled_alias1,DOUBLE,1635201377,0,GOOD,5.0
unmodeled_alias1,DOUBLE,1635201378,0,GOOD,6.0
unmodeled_alias1,DOUBLE,1635201379,0,GOOD,7.0
unmodeled_alias1,DOUBLE,1635201380,0,GOOD,8.0
unmodeled_alias1,DOUBLE,1635201381,0,GOOD,9.0
unmodeled_alias1,DOUBLE,1635201382,0,GOOD,10.0
```

AWS IoT SiteWise provides the following API operations to create a bulk import job and get information about an existing job.

- CreateBulkImportJob – Creates a new bulk import job.
- DescribeBulkImportJob – Retrieves information about a bulk import job.
- ListBulkImportJob – Retrieves a paginated list of summaries of all bulk import jobs.

Create a bulk import job (AWS CLI)

Use the CreateBulkImportJob API operation to transfer data from Amazon S3 to AWS IoT SiteWise. Use the CreateBulkImportJob API to ingest data in small batches in a cost effective way. The following example uses AWS CLI.

> **Important**
>
> Before creating a bulk import job, you must enable AWS IoT SiteWise warm tier or AWS IoT SiteWise cold tier. For more information, see Configure storage settings.
> Bulk import is designed to store historical data to AWS IoT SiteWise. It does not start computations or notifications on AWS IoT SiteWise warm tier or AWS IoT SiteWise cold tier.

Run the following command. Replace *file-name* with the name of the file that contains the bulk import job configuration.

```
aws iotsitewise create-bulk-import-job --cli-input-json file://file-name.json
```

Example Bulk import job configuration

The following are examples of configuration settings:

- Replace *adaptive-ingestion-flag* with `true` or `false`.
  - If set to `false`, the bulk import job ingests historical data into AWS IoT SiteWise.
  - If set to `true`, the bulk import job does the following:
    - Ingests new data into AWS IoT SiteWise.
    - Calculates metrics and transforms, and supports notifications for data with a time stamp that's within seven days.

- Replace *delete-files-after-import-flag* with `true` to delete the data from the S3 data bucket after ingesting into AWS IoT SiteWise warm tier storage.
- Replace *error-bucket* with the name of the Amazon S3 bucket to which errors associated with this bulk import job are sent.
- Replace *error-bucket-prefix* with the prefix of the Amazon S3 bucket to which errors associated with this bulk import job are sent.

  Amazon S3 uses the prefix as a folder name to organize data in the bucket. Each Amazon S3 object has a key that is its unique identifier in the bucket. Each object in a bucket has exactly one key. The prefix must end with a forward slash (/). For more information, see Organizing objects using prefixes in the *Amazon Simple Storage Service User Guide*.
- Replace *data-bucket* with the name of the Amazon S3 bucket from which data is imported.
- Replace *data-bucket-key* with the key of the Amazon S3 object that contains your data. Each object has a key that is a unique identifier. Each object has exactly one key.
- Replace *data-bucket-version-id* with the version ID to identify a specific version of the Amazon S3 object that contains your data. This parameter is optional.
- Replace *column-name* with the column name specified in the .csv file.
- Replace *job-name* with a unique name that identifies the bulk import job.
- Replace *job-role-arn* with the IAM role that allows AWS IoT SiteWise to read Amazon S3 data.

Note

Make sure that your role has the permissions shown in the following example. Replace *data-bucket* with the name of the Amazon S3 bucket that contains your data. Also, replace *error-bucket* with the name of the Amazon S3 bucket to which errors associated with this bulk import job are sent.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:GetBucketLocation"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::data-bucket",
                "arn:aws:s3:::data-bucket/*",
            ],
            "Effect": "Allow"
        },
       {
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::error-bucket",
                "arn:aws:s3:::error-bucket/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

```
{
   "adaptiveIngestion": adaptive-ingestion-flag,
   "deleteFilesAfterImport": delete-files-after-import-flag,
   "errorReportLocation": {
      "bucket": "error-bucket",
      "prefix": "error-bucket-prefix"
   },
   "files": [
      {
         "bucket": "data-bucket",
         "key": "data-bucket-key",
         "versionId": "data-bucket-version-id"
      }
   ],
   "jobConfiguration": {
      "fileFormat": {
         "csv": {
            "columnNames": [ "column-name" ]
         }
      }
   },
```

```
   "jobName": "job-name",
   "jobRoleArn": "job-role-arn"
}
```

Example response

```
{
   "jobId":"f8c031d0-01d1-4b94-90b1-afe8bb93b7e5",
   "jobStatus":"PENDING",
   "jobName":"myBulkImportJob"
}
```

Describe a bulk import job (AWS CLI)

Use the DescribeBulkImportJob API operation to retrieve information about a bulk import job. The following example uses AWS CLI.

Replace *job-ID* with the ID of the bulk import job that you want to retrieve.

```
aws iotsitewise describe-bulk-import-job --job-id job-ID
```

Example response

```
{
   "files":[
      {
         "bucket":"test-bucket",
         "key":"100Tags12Hours.csv"
      },
      {
         "bucket":"test-bucket",
         "key":"BulkImportData1MB.csv"
      },
      {
         "bucket":"test-bucket",
         "key":"UnmodeledBulkImportData1MB.csv"
      }
   ],
   "errorReportLocation":{
      "prefix":"errors/",
      "bucket":"test-error-bucket"
```

```
   },
   "jobConfiguration":{
      "fileFormat":{
         "csv":{
            "columnNames":[
               "ALIAS",
               "DATA_TYPE",
               "TIMESTAMP_SECONDS",
               "TIMESTAMP_NANO_OFFSET",
               "QUALITY",
               "VALUE"
            ]
         }
      }
   },
   "jobCreationDate":1645745176.498,
   "jobStatus":"COMPLETED",
   "jobName":"myBulkImportJob",
   "jobLastUpdateDate":1645745279.968,
   "jobRoleArn":"arn:aws:iam::123456789012:role/DemoRole",
   "jobId":"f8c031d0-01d1-4b94-90b1-afe8bb93b7e5"
}
```

List bulk import jobs (AWS CLI)

Use the ListBulkImportJobs API operation to retrieve a paginated list of summaries of all bulk import jobs. The following example uses AWS CLI.

```
aws iotsitewise list-bulk-import-jobs --filter COMPLETED
```

Example response

```
{
   "jobSummaries":[
      {
         "id":"bdbbfa52-d775-4952-b816-13ba1c7cb9da",
         "name":"myBulkImportJob",
         "status":"COMPLETED"
      },
      {
         "id":"15ffc641-dbd8-40c6-9983-5cb3b0bc3e6b",
         "name":"myBulkImportJob2",
```

```
         "status":"RUNNING"
      }
   ]
}
```

Using SiteWise Edge gateways

An AWS IoT SiteWise Edge gateway serves as the intermediary between your industrial equipment and AWS IoT SiteWise. The SiteWise Edge gateway runs on AWS IoT Greengrass V2 that supports data collection and processing on premises. You can use AWS OpsHub for AWS IoT SiteWise to manage your SiteWise Edge gateways and monitor on-site operations.

You can monitor data locally in your facility using SiteWise Monitor portals on your local devices. For more information, see Enabling your portal at the edge.

Topics

SiteWise Edge gateway requirements

AWS IoT SiteWise Edge gateways run on AWS IoT Greengrass V2 as a set of AWS IoT Greengrass components that support data collection, processing, and publishing on premises. To configure a

SiteWise Edge gateway that runs on AWS IoT Greengrass V2, you must create a gateway in the AWS Cloud and run the SiteWise Edge gateway software to set up your local device.

Requirements

Local devices must meet the following requirements to install and run the SiteWise Edge gateway software.

- Supports AWS IoT Greengrass V2 Core software version v2.3.0 or newer. For more information, see Requirements in the *AWS IoT Greengrass Version 2 Developer Guide*.
- One of the following supported platforms:
  - OS: Ubuntu 20.04 or later

    Architecture: x86_64 (AMD64) or ARMv8 (Aarch64)
  - OS: Red Hat Enterprise Linux (RHEL) 8

    Architecture: x86_64 (AMD64) or ARMv8 (Aarch64)
  - OS: Amazon Linux 2

    Architecture: x86_64 (AMD64) or ARMv8 (Aarch64)
  - OS: Debian 11

    Architecture: x86_64 (AMD64) or ARMv8 (Aarch64)
  - OS: Windows Server 2019 and later

    Architecture: x86_64 (AMD64)

  > **Note**
  >
  > ARM platforms support SiteWise Edge gateways with Data Collection Pack only. The Data Processing Pack is not supported.

- Minimum 4 GB RAM.
- Minimum 10 GB disk space available for the SiteWise Edge gateway software.
- If you plan to process data at the edge with AWS IoT SiteWise, your local device must also meet the following requirements:
  - Has an x86 64 bit quad-core processor.
  - Has at least 16 GB of RAM.

  - Has at least 32 GB for RAM if using Windows.
  - Had at least 256 GB of free disk space.
- The minimum disk space and compute capacity requirements depend on a variety of factors that are unique to your implementation and use case.
  - The disk space required for caching data for intermittent internet connectivity depends on the following factors:
    - Number of data streams uploaded
    - Data points per data stream per second
    - Size of each data point
    - Communication speeds
    - Expected network downtime
  - The compute capacity required to poll and upload data depends on the following factors:
    - Number of data streams uploaded
    - Data points per data stream per second
- Configure your local device to make sure that the following ports are accessible:
  - The local device must allow network inbound traffic on port 443.
  - The local device must allow outbound traffic on port 443 and 8883.

    For a full list of the required outbound service endpoints, see Required service endpoints for AWS IoT SiteWise Edge gateways.
  - The following ports are reserved for use by AWS IoT SiteWise: 80, 443, 3001, 4569, 4572, 8000, 8081, 8082, 8084, 8085, 8445, 8086, 9000, 9500, 11080, and 50010. Using a reserved port for traffic can result in a terminated connection.
- Java Runtime Environment (JRE) version 11 or higher. Java must be available on the PATH environment variable on the device. To use Java to develop custom components, you must install a Java Development Kit (JDK). We recommend that you use Amazon Corretto or OpenJDK.

You must have the following permissions to use SiteWise Edge gateways:

> **Note**
>
> If you use the AWS IoT SiteWise console to create your SiteWise Edge gateway, these permissions are added for you.

- The IAM role for your SiteWise Edge gateway must allow you to use an SiteWise Edge gateway on an AWS IoT Greengrass V2 device to process asset model data and asset data.

  The role allows the following service to assume the role: `credentials.iot.amazonaws.com`.

  Permissions details

  The role must have the following permissions:

  - `iotsitewise` – Allows principals to retrieve asset model data and asset data at the edge.
  - `iot` – Allows your AWS IoT Greengrass V2 devices to interact with AWS IoT.
  - `logs` – Allows your AWS IoT Greengrass V2 devices to send logs to Amazon CloudWatch Logs.
  - `s3` – Allows your AWS IoT Greengrass V2 devices to download custom component artifacts from Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:BatchPutAssetPropertyValue",
                "iotsitewise:List*",
                "iotsitewise:Describe*",
                "iotsitewise:Get*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeCertificate",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "iot:Connect",
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive",
```

```
                "iot:DescribeEndpoint"
            ],
            "Resource": "*"
        }
    ]
}
```

Creating a SiteWise Edge gateway

You can use the AWS IoT SiteWise console to create a SiteWise Edge gateway. This procedure details how to create a self-hosted SiteWise Edge gateway that you'll install on your own hardware. For information about creating a SiteWise Edge gateway that runs on Siemens Industrial Edge, see Running SiteWise Edge on Siemens Industrial Edge.

Create a SiteWise Edge gateway

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Edge gateways**.
3. Choose **Create gateway**.
4. For **Deployment type**, choose **Self-hosted gateway**.
5. Enter a name for your SiteWise Edge gateway or use the name generated by AWS IoT SiteWise.
6. Under **Greengrass device OS**, select the operating system of the device where you'll install this SiteWise Edge gateway.

   > **Note**
   >
   > The Data Processing Pack is only available on x86 platforms.

7. (Optional) To process and organize data at the edge, under **Edge capabilities**, select **Data Processing Pack**.

   > **Note**
   >
   > To grant user groups in your corporate directory access to this SiteWise Edge gateway, see Setting up edge capability

8. (Optional) Under **advanced configuration**, do the following:

- For **Greengrass core device**, choose one of the following options:
  - **Default setup** –: AWS automatically uses default settings to create a Greengrass core device in AWS IoT Greengrass V2.
    1. Enter a name for the Greengrass core device or use the name generated by AWS IoT SiteWise.
  - **Advanced setup** –: Choose this option if you want to use an existing Greengrass core device or to create one manually.
    1. Choose a Greengrass core device or choose **Create Greengrass core device** to create one in the AWS IoT Greengrass V2 console. For more information, see Setting up AWS IoT Greengrass V2 core devices in the *AWS IoT Greengrass Version 2 Developer Guide*.

9. Choose **Create gateway**.
10. In the **Generate SiteWise Edge gateway installer** dialog box, choose **Generate and download**. AWS IoT SiteWise automatically generates an installer that you can use to configure your local device.

> **Important**
>
> Make sure that you save the installer file in a secure location. You will use the file later.

Now that you've created the SiteWise Edge gateway, add data sources, configure the publisher component, and get your SiteWise Edge gateway receiving data and sending it to the AWS Cloud.

Installing the SiteWise Edge gateway software on your local device

Once you've created an SiteWise Edge gateway, you need to install the SiteWise Edge gateway software on your local device. SiteWise Edge gateway software can be installed on local devices that have Linux or Windows server operating systems installed.

> **Important**
>
> Make sure that your local device connects to the internet.

Linux

The following procedure uses SSH to connect to your local device. Alternatively, you can use a USB flash drive or other tools to transfer the installer file to your local device. If you don't want to use SSH, skip to **Step 2: Install the SiteWise Edge gateway software** below.

SSH prerequisites

Before you connect to your device using SSH, complete the following prerequisites.

- Get the IP address of your device.
- Get the username to connect to your device.
- Install an SSH client on your local computer as needed.

  Your local computer might have an SSH client installed by default. You can verify this by typing **ssh** in the command line. If your computer doesn't recognize the command, you can install an SSH client.

  - Linux and macOS - Download and install OpenSSH. For more information, see https://www.openssh.com.

Step 1: Copy the installer to your SiteWise Edge gateway device

The following instructions explain how to connect to your local device using an SSH client.

1. To connect to your device, run the following command in a terminal window on your computer, replacing *username* and *IP* with a username that has elevated priveleges and IP address.

   ```
   ssh username@IP
   ```

2. To transfer the installer file that AWS IoT SiteWise generated to your SiteWise Edge gateway device, run the following command.

   Note

   - Replace *path-to-saved-installer* with the path on your computer that you used to save the installer file and the name of the installer file.
   - Replace *IP-address* with the IP address of your local device.

- Replace *directory-to-receive-installer* with the path on your local device that you use to receive the installer file.

```
scp path-to-saved-installer.sh user-name@IP-address:directory-to-receive-installer
```

Step 2: Install the SiteWise Edge gateway software

In the following procedures, run the commands in a terminal window on your SiteWise Edge gateway device.

1. Give the installer file the execute permission.

```
chmod +x path-to-installer.sh
```

2. Run the installer.

```
sudo ./path-to-installer.sh
```

Windows server

Prerequisites

You must have the following prerequisites to install the SiteWise Edge gateway software:

- Windows Server 2019 or later installed
- Administrator privileges
- PowerShell version 5.1 or later installed
- SiteWise Edge gateway installer downloaded to the Windows Server where it will be provisioned

Step 1: Run PowerShell as administrator

1. On the Windows server where you want to install SiteWise Edge gateway, log in as administrator.

2. Enter **PowerShell** in the Windows search bar.
3. In the search results, open the context (right-click) menu on the Windows PowerShell app. Choose **Run as Administrator**.

Step 2: Install the SiteWise Edge gateway software

Run the following commands in a terminal window on your SiteWise Edge Gateway device.

1. Unblock the SiteWise Edge gateway installer.

```
unblock-file path-to-installer.ps1
```

2. Run the Installer.

```
./path-to-installer.ps1
```

> **Note**
>
> If the script execution is disabled on the system, change the script execution policy to `RemoteSigned`.
>
> ```
> Set-ExecutionPolicy RemoteSigned
> ```

Enabling edge data processing

You can use AWS IoT SiteWise Edge to collect, store, organize and monitor equipment data locally. You can use SiteWise Edge so that you can model your industrial data and SiteWise Monitor to create dashboards for your operational staff to visualize data locally. You can process your data locally and send it to the AWS Cloud, or process it on-premises by using the AWS IoT SiteWise API.

With AWS IoT SiteWise Edge, you can process raw data locally and choose to send only aggregated data to the AWS Cloud to optimize your bandwidth usage and cloud storage costs.

Note

- AWS IoT SiteWise retains your edge data on your SiteWise Edge gateways up to 30 days. The retention period of your data is dependent on the available disk space of your device.
- If your SiteWise Edge gateway has been disconnected from the AWS Cloud for 30 days, the Data Processing Pack is automatically disabled.

Setting up edge capability

AWS IoT SiteWise provides the following packs that your SiteWise Edge gateway can use to determine how to collect and process your data. Select packs to enable edge capabilities for your SiteWise Edge gateway.

- **Data Collection Pack** enables your SiteWise Edge gateway to collect data from multiple OPC-UA servers, and then export the data from the edge to the AWS Cloud. It becomes active once you have added data sources to your SiteWise Edge gateway.
- **Data Processing Pack** enables your SiteWise Edge gateway to process your equipment data at the edge. For example, you can use asset models to compute metrics and transforms. For more information about asset models and assets, see Modeling industrial assets.

Note

The Data Processing Pack is only available on x86 platforms.

To configure edge capabilities

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Edge gateways**.
3. Select the SiteWise Edge gateway for which you want to activate edge capabilities.
4. In the **Edge capabilities** section, choose **Edit**
5. In the **Edge capabilities** section, select **Enable data processing pack (incurs additional charges)**.
6. (Optional) In the **Edge LDAP connection** section, you can grant user groups in your corporate directory access to this SiteWise Edge gateway. The user groups can use the Lightweight

Directory Access Protocol (LDAP) credentials to access the SiteWise Edge gateway. Then they can use the AWS OpsHub for AWS IoT SiteWise application, AWS IoT SiteWise API operations, or other tools to manage the SiteWise Edge gateway. For more information, see Managing SiteWise Edge gateways.

> **Note**
>
> You can also use the Linux or Windows credentials to access the SiteWise Edge gateway. For more information, see Accessing your SiteWise Edge gateway using Linux operating system credentials.

a. Select **Activated**.

b. For **Provider name**, enter a name for your LDAP provider.

c. For **Hostname or IP address**, enter the hostname or IP address of your LDAP server.

d. For **Port**, enter a port number.

e. For **Base distinguished name (DN)**, enter a distinguished name (DN) for the base.

   The following attribute types are supported: commonName (CN), localityName (L), stateOrProvinceName (ST), organizationName (O), organizationalUnitName (OU), countryName (C), streetAddress (STREET), domainComponent (DC), and userid (UID).

f. For **Admin group DN**, enter a DN.

g. For **User group DN**, enter a DN.

7. Choose **Save**.

Now that you've activated edge capabilities on your SiteWise Edge gateway, you need to configure your asset model for the edge. Your asset model edge configuration specifies where your assets properties are computed. You can compute all properties at the edge, or you can configure your asset model properties separately. Asset model properties include metrics, transforms, and measurements.

For more information about asset properties, see the section called “Defining data properties”.

After you create your asset model, you can then configure it for the edge. For more information about configuring your asset model for the edge, see the section called “Creating an asset model (console)”.

> **Note**
>
> Asset models and dashboards are automatically synced between the AWS Cloud and your SiteWise Edge gateway every 10 minutes. You can also sync manually from the local SiteWise Edge gateway application.

Processing data at the edge

You must configure your asset model for the edge before your can process your SiteWise Edge gateway data at the edge. Your asset model edge configuration specifies where your assets properties are computed. You can choose to compute all properties at the edge and send the results to the AWS Cloud, or customize where to compute each asset property separately. For more information, see Enabling edge data processing.

Asset properties include metrics, transforms, and measurements:

- Metrics are the asset's aggregated data over a specified period of time. You can compute new metrics by using existing metric data. AWS IoT SiteWise always sends your metrics to the AWS Cloud for long-term storage. AWS IoT SiteWise computes metrics on the AWS Cloud by default. You can configure your asset model to compute your metrics at the edge. AWS IoT SiteWise sends processed results to the AWS Cloud.
- Transforms are mathematical expressions that map an asset property's data points from one form to another. Transforms can use metrics as input data and must be computed and stored at the same location as their inputs. If you configure a metric input to compute at the edge, AWS IoT SiteWise also computes its associated transform at the edge.
- Measurements are formatted as raw data that your device collects and sends to the AWS Cloud by default. You can configure your asset model to store this data on your local device.

For more information about asset properties, see the section called “Defining data properties”.

After you create your asset model, you can then configure it for the edge. For more information about configuring your asset model for the edge, see the section called “Creating an asset model (console)”.

> **Note**
>
> Asset models and dashboards are automatically synced between the AWS Cloud and your SiteWise Edge gateway every 10 minutes. You can also sync manually from the Managing SiteWise Edge gateways.

You can use the AWS IoT SiteWise REST APIs and the AWS Command Line Interface (AWS CLI) to query your SiteWise Edge gateway for data at the edge. Before you query your SiteWise Edge gateway for data at the edge, you must meet the following prerequisites:

- Your credentials must be set for the REST APIs. For more information about setting credentials, see the section called “Managing SiteWise Edge gateways”.
- The SDK endpoint must point to the IP address of your SiteWise Edge gateway. You can find more information in the documentation for your SDK. For example, see Specifying Custom Endpoints in the *AWS SDK for Java 2.x Developer Guide*.
- Your SiteWise Edge gateway certificate must be registered. You can find more information about registering your SiteWise Edge gateway certificate in the documentation for your SDK. For example, see the Registering Certificate Bundles in Node.js in the *AWS SDK for Java 2.x Developer Guide*.

For more information about querying data with AWS IoT SiteWise, see *Query data from AWS IoT SiteWise*.

Configuring the AWS IoT SiteWise Publisher component

After you create an AWS IoT SiteWise Edge gateway and install the software, set up the Publisher component so your SiteWise Edge gateway can export data to the AWS Cloud. For more information, see AWS IoT SiteWise Publisher in the *AWS IoT Greengrass Version 2 Developer Guide*.

Console

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Edge gateways**.
3. Select the SiteWise Edge gateway for which you want to configure the publisher.
4. In the **Publisher configuration** section, choose **Edit**

5. For **Publishing order**, choose one of the following:

   - **Publish oldest data first** – The SiteWise Edge gateway publishes the earliest data to the cloud first by default.
   - **Publish newest data first** – The SiteWise Edge gateway publishes the newest data to the cloud first.

6. (Optional) If you don't want the SiteWise Edge gateway to compress your data, unselect **Activate compression when uploading data**.
7. (Optional) If you don't want to publish old data, choose **Exclude expired data** and do the following:

   - For **Cutoff period**, enter a number and choose a unit. The cutoff period must be between five minutes and seven days. For example, if the cutoff period is three days, data that's earlier than three days isn't published to the cloud.

8. (Optional) To set custom settings about how data is handled on your local device, choose **Local storage settings** and do the following:

   a. For **Retention period**, enter a number and choose a unit. The retention period must be between one minute and 30 days, and greater than or equal to the rotation period. For example, if the retention period is 14 days, the SiteWise Edge gateway deletes any data at the edge that's earlier than the specified cutoff period after it's stored for 14 days.

   b. For **Rotation period**, enter a number and choose a unit. The rotation period must be greater than one minute, and equal to, or less than, the retention period. For example, if the rotation period is two days, the SiteWise Edge gateway batches up and saves data that is earlier than the cutoff period to a single file. The SiteWise Edge gateway also transfers a batch of data to the following local directory once every two days: `/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports`.

   c. For **Storage capacity**, enter a number that is greater than or equal to 1. If the storage capacity is 2 GB, the SiteWise Edge gateway starts deleting data when more than 2 GB of data is stored locally.

9. Choose **Save**.

AWS CLI

You can use the UpdateGatewayCapabilityConfiguration API to configure the publisher. Set the `capabilityNamespace` parameter to `iotsitewise:publisher:2`.

The publisher provides the following configuration parameters that you can customize:

`SiteWisePublisherConfiguration`

`publishingOrder`

The order in which data is published to the cloud. The value of this parameter can be one of the following:

- `TIME_ORDER` (**Publish oldest data first**) – The earliest data is published to the cloud first, by default.
- `RECENT_DATA` (**Publish newest data first**) – The newest data is published to the cloud first.

`dropPolicy`

(Optional) A policy that controls what data is published to the cloud.

`cutoffAge`

Data that is earlier than the cutoff period isn't published to the cloud. The cutoff age must be between five minutes and seven days.

You can use `m`, `h`, and `d` when you specify a cutoff age. Note that `m` represents minutes, `h` represents hours, and `d` represents days.

`exportPolicy`

(Optional) A policy that manages data storage at the edge. This policy applies to data that is earlier than the cutoff age.

`retentionPeriod`

Your SiteWise Edge gateway deletes any data at the edge that is earlier than the cutoff period from the local storage after it's stored for the specified retention period. The retention period must be between one minute and 30 days, and greater than or equal to the rotation period.

You can use `m`, `h`, and `d` when you specify a retention period. Note that `m` represents minutes, `h` represents hours, and `d` represents days.

`rotationPeriod`

The time interval over which to batch up and save data that is earlier than the cutoff period to a single file. The SiteWise Edge gateway transfers one batch

of data to the following local directory at the end of each rotation period: `/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports`. The rotation period must be greater than one minute, and equal to or less than the retention period.

You can use `m`, `h`, and `d` when you specify a rotation period. Note that `m` represents minutes, `h` represents hours, and `d` represents days.

`exportSizeLimitGB`

The maximum allowed size of data stored locally, in GB. If this quota is breached, the SiteWise Edge gateway starts deleting the earliest data until the size of data stored locally is equal to or less than the quota. The value of this parameter must be greater than or equal to 1.

`SiteWiseS3PublisherConfiguration`

`accessRoleArn`

The access role that gives AWS IoT SiteWise permission to manage the Amazon S3 bucket that you are publishing to.

`streamToS3ConfigMapping`

An array of configurations that maps a stream to an Amazon S3 configuration.

`streamName`

The stream to read from and publish to the Amazon S3 configuration.

`targetBucketArn`

The bucket ARN to publish to.

`publishPolicy`

`publishFrequency`

The frequency with which the SiteWise Edge gateway publishes to the Amazon S3 bucket.

`localSizeLimitGB`

The maximum size of the files written to local disk. If this threshold is breached, the publisher publishes all buffered data to its destination.

`siteWiseImportPolicy`

`enableSiteWiseStorageImport`

Set this to `true` to import data from an Amazon S3 bucket to AWS IoT SiteWise storage.

`enableDeleteAfterImport`

Set this to `true` to delete the file in the Amazon S3 bucket after ingestion into the AWS IoT SiteWise storage.

Example publisher configuration:

The publisher namespace: `iotsitewise:publisher:2`

```
{
"SiteWisePublisherConfiguration": {
    "publishingOrder": "TIME_ORDER",
    "dropPolicy": {
        "cutoffAge": "7d",
        "exportPolicy": {
            "retentionPeriod": "7d",
            "rotationPeriod": "6h",
            "exportLocation": "/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/
exports",
            "exportSizeLimitGB": 10
        }
    }
},
"SiteWiseS3PublisherConfiguration": {
    "accessRoleArn": "arn:aws:iam:123456789012:role/roleName",
    "streamToS3ConfigMapping": [
        {
            "streamName": "S3_OPC-UA_Data_Collector",
            "targetBucketArn": "arn:aws:s3:::myBucket/dataCollector",
            "publishPolicy": {
                "publishFrequency": "10m",
                "localSizeLimitGB": 10
            },
            "siteWiseImportPolicy": {
                "enableSiteWiseStorageImport": true,
                "enableDeleteAfterImport": true
            }
        }
```

```
    ]
}
}
```

Required SiteWise Edge gateway permissions

The permissions attached to your local device must be updated to support buffered ingestion.

- Update the permissions of the AWS IoT SiteWise cloud role. This role is specified in the publisher capability configuration. The role must have the following permissions:

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:DeleteObject"
        ],
        "Resource": [
            "targetBucketArn_1",
            "targetBucketArn_1/*",
             "targetBucketArn_2",
             "targetBucketArn_2"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketLocation"
        ],
        "Resource": [
            "targetBucketArn_1",
            "targetBucketArn_1/*",
             "targetBucketArn_2",
             "targetBucketArn_2"
        ]
    },
    {
        "Effect": "Allow",
```

```
        "Action": [
            "s3:PutObject"
        ],
        "Resource": [
            "targetBucketArn_1/errorreports",
            "targetBucketArn_1/errorreports/*",
            "targetBucketArn_2/S3PutAccessTestKey",
            "targetBucketArn_2/errorreports",
            "targetBucketArn_2/errorreports/*",
            "targetBucketArn_2/S3PutAccessTestKey"
        ]
    }
]
}
```

- Update the permissions associated with your SiteWise Edge gateway IAM role. See [(Optional) Update IAM permissions on your SiteWise Edge gateway].

(Optional) Update IAM permissions on your SiteWise Edge gateway

If you are using a manually created greengrass core device for this SiteWise Edge gateway, and wish to publish data to Amazon S3, you must manually configure the IAM permissions associated with your SiteWise Edge gateway.

The role must have the following permissions:

- s3 – Allow your AWS IoT Greengrass V2 publisher to publish data to the Amazon S3 bucket locations specified by your data sources and publisher.

```
{
"Effect": "Allow",
"Action": [
    "s3:PutObject"
],
"Resource": [
    "targetBucketArn_1/",
    "targetBucketArn_1/*",
    "targetBucketArn_2/",
    "targetBucketArn_2/*"
]
```

```
}
```

- `iotsitewise` – Allow your AWS IoT Greengrass V2 publisher to call the CreateBulkImportJob API provided by AWS IoT SiteWise. This imports the published files into AWS IoT SiteWise warm tier.

```
{
"Effect": "Allow",
"Action": [
    "iotsitewise:CreateBulkImportJob"
],
"Resource": "*"
}
```

- `iam` – Allow your AWS IoT Greengrass V2 publisher to pass the access role that gives AWS IoT SiteWise permission to manage your Amazon S3 buckets to which you are publishing. This role is created by the AWS IoT SiteWise console when you save a data source on your SiteWise Edge gateway with the **AWS IoT SiteWise Buffered using Amazon S3** destination. You can access it from your account IAM roles. To find it, filter roles by your SiteWise Edge gateway ID. The role has the naming format `IoTSiteWise-S3-Ingest-<gatewayId>`. You can also find it in your publisher configuration under:

  `SiteWiseS3PublisherConfiguration` – `accessRoleArn`

```
{
"Effect": "Allow",
"Action": [
    "iam:PassRole"
],
"Resource": [
    "roleArn"
]
}
```

Configuring data sources

After you set up an AWS IoT SiteWise Edge gateway, you can configure data sources so that your SiteWise Edge gateway can ingest data from local industrial equipment to AWS IoT SiteWise.

Each source represents a local server, such as an OPC-UA server, that your SiteWise Edge gateway connects and retrieves industrial data streams. For more information about setting up a SiteWise Edge gateway, see Configuring a AWS IoT Greengrass V1 SiteWise Edge gateway.

> **Note**
>
> AWS IoT SiteWise restarts your SiteWise Edge gateway each time you add or edit a source. Your SiteWise Edge gateway won't ingest data while it's restarting. The time to restart your SiteWise Edge gateway depends on the number of tags on your SiteWise Edge gateway's sources. Restart time can range from a few seconds (for a SiteWise Edge gateway with few tags) to several minutes (for a SiteWise Edge gateway with many tags).

After you create sources, you can associate your data streams with asset properties. For more information about how to create and use assets, see Modeling industrial assets and Mapping industrial data streams to asset properties.

You can view CloudWatch metrics to verify that a data source is connected to AWS IoT SiteWise. For more information, see AWS IoT Greengrass Version 2 gateway metrics.

Currently, AWS IoT SiteWise supports the following data source protocols:

- OPC-UA – A machine-to-machine (M2M) communication protocol for industrial automation.

> **Note**
>
> SiteWise Edge gateways running on AWS IoT Greengrass V2 currently don't support Modbus TCP and Ethernet IP sources.

Topics

- Configure an OPC-UA source
- Configuring data source authentication
- Choosing a destination for your source server data

Configure an OPC-UA source

You can use the AWS IoT SiteWise console or a SiteWise Edge gateway capability to define and add an OPC-UA source to your SiteWise Edge gateway to represent a local OPC-UA server.

Topics

Configure an OPC-UA source (console)

To configure an OPC-UA source using the AWS IoT SiteWise console

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Gateways**.
3. Select the SiteWise Edge gateway to add an OPC-UA source.
4. Choose **Add data source**.
5. Enter a name for the source.
6. (Optional) Enter a **Data stream prefix**. The SiteWise Edge gateway adds this prefix to all data streams from this source. Use a data stream prefix to distinguish between data streams that have the same name from different sources. Each data stream should have a unique name within your account.
7. Enter the **Local endpoint** of the data source server. The endpoint can be the IP address or hostname. You may also add a port number to the local endpoint. For example, your local endpoint might look like this: **`opc.tcp://203.0.113.0:49320`**

   > **Note**
   >
   > If your SiteWise Edge gateway has a `Deployment type` of **Siemens Industrial Edge device - new** and you want to ingest data from the Edge OPC UA Server application running on the same Siemens Industrial Edge Device as the AWS IoT SiteWise Edge application, enter **`opc.tcp://ie-opcua:48010`**.

8. (Optional) For **Node ID for selection**, add node filters to limit which data streams are ingested to the AWS Cloud. By default, SiteWise Edge gateways use the root node of a server to ingest all data streams. To define node filters, you can use node IDs and the * and ** wildcard characters.
9. For **Destinations**, choose the destination for the source data:

   - **AWS IoT SiteWise real-time** – Choose this to send data directly to AWS IoT SiteWise storage. Ingest and monitor data in real-time, and process data at the edge.
   - **AWS IoT SiteWise Buffered using Amazon S3** – Send data in parquet format to Amazon S3 and then import into AWS IoT SiteWise storage. Choose this option to ingest data in batches, and store historical data in a cost-effective way. You can configure your preferred Amazon S3 bucket location, and the frequency at which you want data to be uploaded to Amazon S3. You can also choose what to do with the data after ingestion into AWS IoT SiteWise. You can choose to have the data available in both SiteWise and Amazon S3 or you can choose to delete it automatically from Amazon S3.
     - The Amazon S3 bucket is a staging and buffering mechanism and supports files in the parquet format.
     - If you select the check box **Import data into AWS IoT SiteWise storage**, data is uploaded into Amazon S3 first, and then into AWS IoT SiteWise storage.
       - If you select the check box **Delete data from Amazon S3**, data is deleted from Amazon S3, after it is imported into SiteWise storage.
       - If you clear the check box **Delete data from Amazon S3**, data is stored both in Amazon S3, and in SiteWise storage.
     - If you clear the check box **Import data into AWS IoT SiteWise storage**, data is stored only in Amazon S3. It is not imported into SiteWise storage.

     Visit Managing data storage for details on the various storage options AWS IoT SiteWise provides. To learn more about pricing options, see AWS IoT SiteWise pricing.
   - **AWS IoT Greengrass stream manager** – Use AWS IoT Greengrass stream manager to send data to the following AWS Cloud destinations: channels in AWS IoT Analytics, streams in Amazon Kinesis Data Streams, asset properties in AWS IoT SiteWise, or objects in Amazon Simple Storage Service (Amazon S3). For more information, see Manage data streams on the AWS IoT Greengrass Core in *AWS IoT Greengrass Version 2 Developer Guide*.

     Enter a name for the AWS IoT Greengrass stream.

10. While configuring a data source, **Node ID for selection** is used to determine the destination of the data flow.

    - If the same data is published to both **AWS IoT SiteWise real-time** and **AWS IoT SiteWise Buffered using Amazon S3**, you must add two data sources that publish to both destinations.
    - To split the data so that a part of it is published to **AWS IoT SiteWise real-time**, and the other part to **AWS IoT SiteWise Buffered using Amazon S3**, you must filter for the following data aliases:

      ```
      /Alias01/Data1
      /Alias02/Data1
      /Alias03/Data1
      /Alias03/Data2
      ```

      For example, you can add a data source pointing to `/**/Data1` node filter, to **AWS IoT SiteWise real-time**, and another data source pointing to `/**/Data2` **AWS IoT SiteWise buffered using Amazon S3**

11. In the **Advanced configuration** pane, do the following:

    a. Choose a **Message security mode** for connections and data in transit between your source server and your SiteWise Edge gateway. This field is the combination of the OPC-UA security policy and message security mode. Choose the same security policy and message security mode that you specified for your OPC-UA server.

    b. If your source requires authentication, choose an AWS Secrets Manager secret from the **Authentication configuration** list. The SiteWise Edge gateway uses the authentication credentials in this secret when it connects to this data source. You must attach secrets to your SiteWise Edge gateway's AWS IoT Greengrass component to use them for data source authentication. For more information, see the section called “Configuring data source authentication”.

       > **Tip**
       >
       > Your data server might have an option named **Allow anonymous login**. If this option is **Yes**, then your source doesn't require authentication.

    c. For **Property groups**, choose **Add new group**.

d. Enter a **Name** for the property group.

e. For **Properties**:

1. (Optional) For **Node paths**, add OPC-UA node filters to limit which OPC-UA paths are uploaded to AWS IoT SiteWise. You can use node filters to reduce your SiteWise Edge gateway's startup time and CPU usage by only including paths to data that you model in AWS IoT SiteWise. By default, SiteWise Edge gateways upload all OPC-UA paths except those that start with `/Server/`. To define OPC-UA node filters, you can use node paths and the * and ** wildcard characters. For more information, see Using OPC-UA node filters.

f. For **Group settings**, do the following:

1. (Optional) For **Data quality setting**, choose the type of data quality that you want AWS IoT SiteWise Collector to ingest.
2. (Optional) For **Scan mode setting**, configure following standard subscription properties:
   - For **Scan mode**, choose the mode that you want AWS IoT SiteWise to use to collect your data. For more information about scan mode, see the section called “Filtering data ingestion ranges with OPC-UA”.
   - Data change initiation –: You can define the condition that initiates a data change alert.
   - Subscription queue size –: The depth of the queue on an OPC–UA server for a particular metric where notifications for monitored items are queued.
   - Subscription publishing interval –: The interval (in milliseconds) of publishing cycle specified when subscription is created.
   - Snapshot interval –: You can configure the snapshot frequency timeout setting to ensure that AWS IoT SiteWise Edge ingests a steady stream of data.
   - For **Scan rate**, update the rate that you want the SiteWise Edge gateway to read your registers. AWS IoT SiteWise automatically calculates the minimum allowable scan rate for your SiteWise Edge gateway.
3. (Optional) Configure a **Deadband type** for your source. This controls what data your source sends to your AWS IoT SiteWise, and what data it discards. For more information about the deadband setting, see the section called “Filtering data ingestion ranges with OPC-UA”.

g. Choose **Add**.

12. Choose **Next**.

Configure an OPC-UA source (CLI)

You can define OPC-UA data sources for an SiteWise Edge gateway using the AWS CLI. To do this, create an OPC-UA capability configuration JSON file and use the update-gateway-capability-configuration command to update the SiteWise Edge gateway configuration. You must define all of your OPC-UA sources in a single capability configuration.

For more information about defining sources with the AWS Command Line Interface, see the section called "Configuring data sources (AWS CLI)".

This capability has the following versions.

Version	Namespace
1	`iotsitewise:opcuacollector:1`

Request syntax

```
{
  "sources": [
    {
      "name": "string",
      "endpoint": {
        "certificateTrust": {
          "type": "string"
          "certificateBody": "string"
          "certificateChain": "string"
        },
        "endpointUri": "string",
        "securityPolicy": "string",
        "messageSecurityMode": "string",
        "identityProvider": {
          "type": "string",
          "usernameSecretArn": "string"
        },
        "nodeFilterRules": [
          {
            "action": "string",
```

```
            "definition": {
              "type": "string",
              "rootPath": "string"
            }
          }
        ]
      },
      "measurementDataStreamPrefix": "string"
      "propertyGroups": [
      {
          "name": "string",
          "deadband": {
              "type":"string",
              "value": string,
              "eguMin": string,
              "eguMax": string,
              "timeoutMilliseconds": string
          },
          "scanMode": {
              "type": "string",
              "rate": string
          },
          "nodeFilterRuleDefinitions": [
              {
                  "type": "string",
                  "rootPath": "string"
              }
          ]
      }
    }
  ]
}
```

Request body

sources

A list of OPC-UA source definition structures that each contain the following information:

name

A unique, friendly name for the source.

endpoint

An endpoint structure that contains the following information:

certificateTrust

A certificate trust policy structure that contains the following information:

type

The certificate trust mode for the source. Choose one of the following:

- **`TrustAny`** – The SiteWise Edge gateway trusts any certificate when it connects to the OPC-UA source.
- `X509` – The SiteWise Edge gateway trusts an X.509 certificate when it connects to the OPC-UA source. If you choose this option, you must define `certificateBody` in `certificateTrust`. You can also define `certificateChain` in `certificateTrust`.

certificateBody

(Optional) The body of an X.509 certificate.

This field is required if you choose `X509` for `type` in `certificateTrust`.

certificateChain

(Optional) The chain of trust for an X.509 certificate.

This field is used only if you choose `X509` for `type` in `certificateTrust`.

endpointUri

The local endpoint of the OPC-UA source. For example, your local endpoint might look like `opc.tcp://203.0.113.0:49320`.

securityPolicy

The security policy to use so that you can secure messages that are read from the OPC-UA source. Choose one of the following:

- `NONE` – The SiteWise Edge gateway doesn't secure messages from the OPC-UA source. We recommend that you choose a different security policy. If you choose this option, you must also choose `NONE` for `messageSecurityMode`.
- `BASIC256_SHA256` – The `Basic256Sha256` security policy.

- AES128_SHA256_RSAOAEP – The Aes128_Sha256_RsaOaep security policy.
- AES256_SHA256_RSAPSS – The Aes256_Sha256_RsaPss security policy.
- BASIC128_RSA15 – (Deprecated) The Basic128Rsa15 security policy is deprecated in the OPC-UA specification because it's no longer considered secure. We recommend that you choose a different security policy. For more information, see Basic128Rsa15.
- BASIC256 – (Deprecated) The Basic256 security policy is deprecated in the OPC-UA specification because it's no longer considered secure. We recommend that you choose a different security policy. For more information, see Basic256.

> **Important**
>
> If you choose a security policy other than NONE, you must choose SIGN or SIGN_AND_ENCRYPT for messageSecurityMode. You must also configure your source server to trust the SiteWise Edge gateway. For more information, see Enabling your OPC-UA source servers to trust the SiteWise Edge gateway.

messageSecurityMode

The message security mode to use to secure connections to the OPC-UA source. Choose one of the following:

- NONE – The SiteWise Edge gateway doesn't secure connections to the OPC-UA source. We recommend that you choose a different message security mode. If you choose this option, you must also choose NONE for securityPolicy.
- SIGN – Data in transit between the SiteWise Edge gateway and the OPC-UA source is signed but not encrypted.
- SIGN_AND_ENCRYPT – Data in transit between the gateway and the OPC-UA source is signed and encrypted.

> **Important**
>
> If you choose a message security mode other than NONE, you must choose a securityPolicy other than NONE. You must also configure your source server to trust the SiteWise Edge gateway. For more information, see Enabling your OPC-UA source servers to trust the SiteWise Edge gateway.

identityProvider

An identity provider structure that contains the following information:

type

The type of authentication credentials required by the source. Choose one of the following:

- `Anonymous` – The source doesn't require authentication to connect.
- `Username` – The source requires a user name and password to connect. If you choose this option, you must define `usernameSecretArn` in `identityProvider`.

usernameSecretArn

(Optional) The ARN of an AWS Secrets Manager secret. The SiteWise Edge gateway uses the authentication credentials in this secret when it connects to this source. You must attach secrets to your SiteWise Edge gateway's IoT SiteWise connector to use them for source authentication. For more information, see Configuring data source authentication.

This field is required if you choose `Username` for `type` in `identityProvider`.

nodeFilterRules

A list of node filter rule structures that define the OPC-UA data stream paths to send to the AWS Cloud. You can use node filters to reduce your SiteWise Edge gateway's startup time and CPU usage by only including paths to data that you model in AWS IoT SiteWise. By default, SiteWise Edge gateways upload all OPC-UA paths except those that start with `/Server/`. To define OPC-UA node filters, you can use node paths and the * and ** wildcard characters. For more information, see Using OPC-UA node filters.

Each structure in the list must contain the following information:

action

The action for this node filter rule. You can choose the following option:

- `INCLUDE` – The SiteWise Edge gateway includes only data streams that match this rule.

definition

A node filter rule structure that contains the following information:

type

The type of node filter path for this rule. You can choose the following option:

- `OpcUaRootPath` – The SiteWise Edge gateway evaluates this node filter path against the root of the OPC-UA path hierarchy.

rootPath

The node filter path to evaluate against the root of the OPC-UA path hierarchy. This path must start with `/`.

measurementDataStreamPrefix

A string to prepend to all data streams from the source. The SiteWise Edge gateway adds this prefix to all data streams from this source. Use a data stream prefix to distinguish between data streams that have the same name from different sources. Each data stream should have a unique name within your account.

propertyGroups

(Optional) The list of property groups that define the `deadband` and `scanMode` requested by the protocol.

name

The name of the property group. This should be a unique identifier.

deadband

The `deadband` structure that contains the following information:

type

The supported types of deadband. Accepted values are `ABSOLUTE` and `PERCENT`.

value

The value of the deadband. When `type` is `ABSOLUTE`, this value is a unitless double. When `type` is `PERCENT`, this value is a double between `1` and `100`.

eguMin

(Optional) The engineering unit minimum when using a `PERCENT` deadband. You set this if the OPC-UA server doesn't have engineering units configured.

eguMax

(Optional) The engineering unit maximum when using a `PERCENT` deadband. You set this if the OPC-UA server doesn't have engineering units configured.

timeoutMilliseconds

The duration in milliseconds before timeout. The minimum is `100`.

scanMode

The `scanMode` structure that contains the following information:

type

The supported types of `scanMode`. Accepted values are `POLL` and `EXCEPTION`.

rate

The sampling interval for the scan mode.

nodeFilterRuleDefinitions

(Optional) A list of node paths to include in the property group. Property groups can't overlap. If you don't specify a value for this field, the group contains all paths under the root, and you can't create additional property groups. The `nodeFilterRuleDefinitions` structure contains the following information:

type

`OpcUaRootPath` is the only supported type. This specifies that the value of `rootPath` is a path relative to the root of the OPC-UA browsing space.

rootPath

A comma-delimited list that specifies the paths (relative to the root) to include in the property group.

Capability configuration examples

The following example defines an OPC-UA SiteWise Edge gateway capability configuration from a payload stored in a JSON file.

```
aws iotsitewise update-gateway-capability-configuration \
--capability-namespace "iotsitewise:opcuacollector:1" \
--capability-configuration file://opc-ua-configuration.json
```

Example : OPC-UA source configuration

The following `opc-ua-configuration.json` file defines a basic, insecure OPC-UA source configuration.

```
{
"sources": [
  {
    "name": "Wind Farm #1",
    "endpoint": {
      "certificateTrust": {
        "type": "TrustAny"
      },
      "endpointUri": "opc.tcp://203.0.113.0:49320",
      "securityPolicy": "NONE",
      "messageSecurityMode": "NONE",
      "identityProvider": {
        "type": "Anonymous"
      },
      "nodeFilterRules": []
    },
    "measurementDataStreamPrefix": ""
  }
]
}
```

Example : OPC-UA source configuration with defined property groups

The following `opc-ua-configuration.json` file defines a basic, insecure OPC-UA source configuration with defined property groups.

```
{
  "sources": [
      {
          "name": "source1",
          "endpoint": {
              "certificateTrust": {
                  "type": "TrustAny"
              },
              "endpointUri": "opc.tcp://10.0.0.9:49320",
              "securityPolicy": "NONE",
              "messageSecurityMode": "NONE",
              "identityProvider": {
```

```
                "type": "Anonymous"
            },
            "nodeFilterRules": [
                {
                    "action": "INCLUDE",
                    "definition": {
                        "type": "OpcUaRootPath",
                        "rootPath": "/Utilities/Tank"
                    }
                }
            ]
         },
         "measurementDataStreamPrefix": "propertyGroups",
        "propertyGroups": [

            {
                "name": "Deadband_Abs_5",
                "nodeFilterRuleDefinitions": [
                    {
                        "type": "OpcUaRootPath",
                        "rootPath": "/Utilities/Tank/Temperature/TT-001"
                    },
                    {
                        "type": "OpcUaRootPath",
                        "rootPath": "/Utilities/Tank/Temperature/TT-002"
                    }
                ],
                "deadband": {
                    "type":"ABSOLUTE",
                    "value": 5.0,
                    "timeoutMilliseconds": 120000
                }
            },
            {
                "name": "Polling_10s",
                "nodeFilterRuleDefinitions": [
                    {
                        "type": "OpcUaRootPath",
                        "rootPath": "/Utilities/Tank/Pressure/PT-001"
                    }
                ],
                "scanMode": {
                    "type": "POLL",
                    "rate": 10000
```

```
                }
            },
            {
                "name": "Percent_Deadband_Timeout_90s",
                "nodeFilterRuleDefinitions": [
                    {
                        "type": "OpcUaRootPath",
                        "rootPath": "/Utilities/Tank/Flow/FT-*"
                    }
                ],
                "deadband": {
                    "type":"PERCENT",
                    "value": 5.0,
                    "eguMin": -100,
                    "eguMax": 100,
                    "timeoutMilliseconds": 90000
                }
            }
        ]
    }
  ]
}
```

Example : OPC-UA source configuration with properties

The following JSON example for `opc-ua-configuration.json` defines an OPC-UA source configuration with the following properties:

- Trusts any certificate.
- Uses the `BASIC256` security policy to secure messages.
- Uses the `SIGN_AND_ENCRYPT` mode to secure connections.
- Uses authentication credentials stored in a Secrets Manager secret.
- Filters out data streams except those whose path starts with `/WindFarm/2/WindTurbine/`.
- Adds `/Washington` to the start of every data stream path to distinguish between this "Wind Farm #2" and a "Wind Farm #2" in another area.

```
{
"sources": [
  {
    "name": "Wind Farm #2",
```

```
    "endpoint": {
      "certificateTrust": {
        "type": "TrustAny"
      },
      "endpointUri": "opc.tcp://203.0.113.1:49320",
      "securityPolicy": "BASIC256",
      "messageSecurityMode": "SIGN_AND_ENCRYPT",
      "identityProvider": {
        "type": "Username",
        "usernameSecretArn":
 "arn:aws:secretsmanager:region:123456789012:secret:greengrass-windfarm2-auth-1ABCDE"
      },
      "nodeFilterRules": [
        {
          "action": "INCLUDE",
          "definition": {
            "type": "OpcUaRootPath",
            "rootPath": "/WindFarm/2/WindTurbine/"
          }
        }
      ]
    },
    "measurementDataStreamPrefix": "/Washington"
  }
]
}
```

Example

The following JSON example for `opc-ua-configuration.json` defines an OPC-UA source configuration with the following properties:

- Trusts a given X.509 certificate.
- Uses the `BASIC256` security policy to secure messages.
- Uses the `SIGN_AND_ENCRYPT` mode to secure connections.

```
{
"sources": [
  {
    "name": "Wind Farm #3",
    "endpoint": {
```

```
      "certificateTrust": {
        "type": "X509",
        "certificateBody": "-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
 0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
 WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
 EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
 jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
 MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
 WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
 HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
 BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
 k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
 ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
 AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
 KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
 EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
 3rrszlaEXAMPLE=
-----END CERTIFICATE-----",
        "certificateChain": "-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
 0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
 WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
 EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
 jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
 MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
 WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
 HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
 BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
 k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
 ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
 AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
 KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
 EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
 3rrszlaEXAMPLE=
-----END CERTIFICATE-----"
      },
      "endpointUri": "opc.tcp://203.0.113.2:49320",
      "securityPolicy": "BASIC256",
      "messageSecurityMode": "SIGN_AND_ENCRYPT",
      "identityProvider": {
        "type": "Anonymous"
      },
      "nodeFilterRules": []
```

```
    },
    "measurementDataStreamPrefix": ""
  }
]
}
```

Enabling your OPC-UA source servers to trust the SiteWise Edge gateway

If you choose a `messageSecurityMode` other than **None** when configuring your OPC-UA source, you must enable your source servers to trust the AWS IoT SiteWise Edge gateway. The SiteWise Edge gateway generates a certificate that your source server might require. The process varies depending on your source servers. For more information, see the documentation for your servers.

The following procedure outlines the basic steps.

To enable an OPC-UA server to trust the SiteWise Edge gateway

1. Open the interface for configuring your OPC-UA server.
2. Enter the user name and password for the OPC-UA server administrator.
3. Locate **Trusted Clients** in the interface, and then choose **AWS IoT SiteWise Gateway Client**.
4. Choose **Trust**.

Exporting the OPC-UA client certificate

Some OPC-UA servers require access to the OPC-UA client certificate file to trust the SiteWise Edge gateway. If this applies to your OPC-UA servers, you can use the following procedure to export the OPC-UA client certificate from the SiteWise Edge gateway. Then, you can import the certificate on your OPC-UA server.

To export the OPC-UA client certificate file for a source

1. Run the following command to change to the directory that contains the certificate file. Replace *`sitewise-work`* with the local storage path for the *`aws.iot.SiteWiseEdgeCollectorOpcua`* Greengrass work folder and replace *`source-name`* with the name of the data source.

   By default, the Greengrass work folder is */greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua* on Linux and *C:/greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua* on Windows.

```
cd /sitewise-work/source-name/opcua-certificate-store
```

2. The SiteWise Edge gateway's OPC-UA client certificate for this source is in the `aws-iot-opcua-client.pfx` file.

   Run the following command to export the certificate to a `.pem` file called `aws-iot-opcua-client-certificate.pem`.

   ```
   keytool -exportcert -v -alias aws-iot-opcua-client -keystore aws-iot-opcua-client.pfx -storepass amazon -storetype PKCS12 -rfc > aws-iot-opcua-client-certificate.pem
   ```

3. Transfer the certificate file, `aws-iot-opcua-client-certificate.pem`, from the SiteWise Edge gateway to the OPC-UA server.

   To do so, you can use common software such as the `scp` program to transfer the file using the SSH protocol. For more information, see Secure copy on *Wikipedia*.

   > **Note**
   >
   > If your SiteWise Edge gateway is running on Amazon Elastic Compute Cloud (Amazon EC2) and you're connecting to it for the first time, you must configure prerequisites to connect. For more information, see Connect to your Linux instance in the *Amazon EC2 User Guide for Linux Instances*.

4. Import the certificate file, `aws-iot-opcua-client-certificate.pem`, on the OPC-UA server to trust the SiteWise Edge gateway. Steps can vary depending on the source server that you use. Consult the documentation for the server.

Filter data ingestion ranges with OPC-UA

You can control the way you ingest data with an OPC-UA source by using scan mode and deadband ranges. These features let you control what kind of data to ingest, and how and when your server and SiteWise Edge gateway exchange this information.

Control data collection frequency with Scan mode

You can configure your OPC-UA scan mode to control the way you collect data from your OPC-UA source. You can choose subscription or polling mode.

- Subscription mode – The OPC-UA source collects data to send to your SiteWise Edge gateway at the frequency defined by your scan rate. The server only sends data when the value has changed, so this is the maximum frequency your SiteWise Edge gateway receives data.
- Polling mode – Your SiteWise Edge gateway polls the OPC-UA source at a set frequency defined by your scan rate. The server sends data regardless of whether the value has changed, so your SiteWise Edge gateway always receives data at this interval.

> **Note**
>
> The polling mode option overrides your deadband settings for this source.

Filter OPC-UA data ingestion with deadband ranges

You can apply a deadband to your OPC-UA source property groups to filter out and discard certain data instead of sending it to the AWS Cloud. A deadband specifies a window of expected fluctuations in the incoming data values from your OPC-UA source. If the values fall within this window, your OPC-UA server won't send it to the AWS Cloud. You can use deadband filtering to reduce the amount of data you're processing and sending to the AWS Cloud. To learn how to set up OPC-UA sources for your SiteWise Edge gateway, see the section called “Configuring data sources”.

> **Note**
>
> Your server deletes all data that falls inside the window specified by your deadband. You can't recover this discarded data.

Types of deadbands

You can specify two types of deadbands for your OPC-UA server property group. These let you choose how much data is sent to the AWS Cloud, and how much is discarded.

- Percentage – You specify a window using a percentage of expected fluctuation in the measurement value. The server calculates the exact window from this percentage, and sends data to the AWS Cloud that exceeds falls outside the window. For example, specifying a 2% deadband value on a sensor with a range from -100 degrees Fahrenheit to +100 degrees Fahrenheit tells the server to send data to the AWS Cloud when the value changes by 4 degrees Fahrenheit or more.

> **Note**
>
> You can optionally specify a minimum and maximum deadband value for this window if your source server doesn't define engineering units. If an engineering unit range is not provided, the OPC-UA server defaults to the full range of the measurement data type.

- Absolute – You specify a window using exact units. For example, specifying a deadband value of 2 on a sensor tells the server to send data to the AWS Cloud when its value changes by at least 2 units. You can use absolute deadbanding for dynamic environments where fluctuations are regularly expected during normal operations.

Deadband timeouts

You can optionally configure a deadband timeout setting. After this timeout, the OPC-UA server sends the current measurement value even if it is within the expected deadband fluctuation. You can use the timeout setting to ensure that AWS IoT SiteWise is ingesting a steady stream of data at all times, even when values do not exceed the defined deadband window.

Using OPC-UA node filters

When you define OPC-UA data sources for an SiteWise Edge gateway, you can define node filters. Node filters let you limit which data stream paths the SiteWise Edge gateway sends to the cloud. You can use node filters to reduce your SiteWise Edge gateway's startup time and CPU usage by only including paths to data that you model in AWS IoT SiteWise. By default, SiteWise Edge gateways upload all OPC-UA paths except those that start with `/Server/`. You can use the * and ** wildcard characters in your node filters to include multiple data stream paths with one filter. To learn how to set up OPC-UA sources for your SiteWise Edge gateway, see Configuring data sources.

> **Note**
>
> AWS IoT SiteWise restarts your SiteWise Edge gateway each time you add or edit a source. Your SiteWise Edge gateway won't ingest data while it's restarting. The time to restart your SiteWise Edge gateway depends on the number of tags on your SiteWise Edge gateway's sources. Restart time can range from a few seconds (for a SiteWise Edge gateway with few tags) to several minutes (for a SiteWise Edge gateway with many tags).

The following table lists the wildcards that you can use to filter OPC-UA data sources.

OPC-UA node filter wildcards

Wildcard	Description
*	Matches a single level in a data stream path.
**	Matches multiple levels in a data stream path.

> **Note**
>
> If you configure a source with a broad filter and then later change the source to use a more restrictive filter, AWS IoT SiteWise stops storing data that doesn't match the new filter.

Example Example scenario using node filters

Consider the following hypothetical data streams:

- `/WA/Factory 1/Line 1/PLC1`
- `/WA/Factory 1/Line 1/PLC2`
- `/WA/Factory 1/Line 2/Counter1`
- `/WA/Factory 1/Line 2/PLC1`
- `/OR/Factory 1/Line 1/PLC1`
- `/OR/Factory 1/Line 2/Counter2`

Using the previous data streams, you can define node filters to limit what data to include from your OPC-UA source.

- To select all nodes in this example, use `/` or `/**/`. You can include multiple directories or folders with the `**` wildcard characters.
- To select all PLC data streams, use `/*/*/*/PLC*` or `/**/PLC*`.
- To select all counters in this example, use `/**/Counter*` or `/*/*/*/Counter*`.
- To select all counters from `Line 2`, use `/**/Line 2/Counter*`.

Configuring data source authentication

If your OPC-UA server requires authentication credentials to connect, you can use AWS Secrets Manager to create and deploy a secret to your SiteWise Edge gateway. AWS Secrets Manager encrypts secrets on the device to keep your user name and password secure until you need to use them. For more information, see Secret manager in the *AWS IoT Greengrass Version 2 Developer Guide*.

Step 1: Create source authentication secrets

You can use AWS Secrets Manager to create an authentication secret for your data source. In the secret, define **`username`** and **`password`** key-value pairs that contain authentication details for your data source.

To create a secret (console)

1. Navigate to the AWS Secrets Manager console.
2. Choose **Store a new secret**.
3. Under **Secret type**, choose **Other type of secrets**.
4. Under **Key/value pairs**, do the following:
    1. In the first input box, enter **`username`** and in the second input box enter the username.
    2. Choose **Add row**.
    3. In the first input box, enter **`password`** and in the second input box enter the password.
5. For **Encryption key**, select **aws/secretsmanager**, and then choose **Next**.
6. On the **Store a new secret** page, enter a **Secret name**.
7. (Optional) Enter a **Description** that helps you identify this secret, and then choose **Next**.
8. (Optional) On the **Store a new secret** page, turn on **Automatic rotation**. For more information, see Rotate secrets in the *AWS Secrets Manager User Guide*.
9. Specify a rotation schedule.
10. Choose a Lambda function that can rotate this secret, and then choose **Next**.
11. Review your secret configurations, and then choose **Store**.

To authorize your SiteWise Edge gateway to interact with AWS Secrets Manager, the IAM role for your SiteWise Edge gateway must allow the `secretsmanager:GetSecretValue` action. You can

use the **Greengrass core device** to search for the IAM policy. For more information about updating an IAM policy, see Editing IAM policies in the *AWS Identity and Access Management User Guide*.

Example policy

Replace *secret-arn* with the Amazon Resource Name (ARN) of the secret that you created in the previous step. For more information about how to get the ARN of a secret, see Retrieve your secret from AWS Secrets Manager in the *AWS Secrets Manager User Guide*.

```
{
"Version":"2012-10-17",
"Statement":[
  {
     "Action":[
        "secretsmanager:GetSecretValue"
     ],
     "Effect":"Allow",
     "Resource":[
        "secret-arn"
     ]
  }
]
}
```

Step 2: Deploy secrets to your SiteWise Edge gateway device

You can use the AWS IoT SiteWise console to deploy secrets to your SiteWise Edge gateway.

To deploy a secret (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Gateways**.
3. From the **Gateways** list, choose the target SiteWise Edge gateway.
4. In the **Gateway configuration** section, choose the **Greengrass core device** link to open the AWS IoT Greengrass core associated with the SiteWise Edge gateway.
5. In the navigation pane, choose **Deployments**.
6. Choose the target deployment, and then choose **Revise**.
7. On the **Specify target** page, choose **Next**.

8. On the **Select components** page, in the **Public components** section, turn off **Show only selected components**.
9. Search for and choose the **aws.greengrass.SecretManager** component, and then choose **Next**.
10. From the **Selected components** list, choose the **aws.greengrass.SecretManager** component, and then choose **Configure component**.
11. In the **Configuration to merge** field, add the following JSON object.

> **Note**
>
> Replace *secret-arn* with the ARN of the secret that you created in the previous step. For more information about how to get the ARN of a secret, see Retrieve your secret from AWS Secrets Manager in the *AWS Secrets Manager User Guide*.

```
{
"cloudSecrets":[
  {
     "arn":"secret-arn"
  }
]
}
```

12. Choose **Confirm**.
13. Choose **Next**.
14. On the **Configure advanced settings** page, choose **Next**.
15. Review your deployment configurations, and then choose **Deploy**.

Step 3: Add authentication configurations

You can use the AWS IoT SiteWise console to add authentication configurations to your SiteWise Edge gateway.

To add authentication configurations (console)

1. Navigate to the AWS IoT SiteWise console.
2. From the **Gateways** list, choose the target SiteWise Edge gateway.
3. From the **Data sources** list, choose the target data source, and then choose **Edit**.

4. On the **Add a data source** page, choose **Advanced configuration**.
5. For **Authentication configuration**, choose the secret that you deployed in the previous step.
6. Choose **Save**.

Configuring data sources (AWS CLI)

You can use the AWS IoT SiteWise API and AWS Command Line Interface to add sources to your AWS IoT SiteWise Edge gateway. You define sources in SiteWise Edge gateway capabilities. A SiteWise Edge gateway capability represents a software feature that runs on the SiteWise Edge gateway, such as the capability to collect industrial data from OPC-UA sources.

SiteWise Edge gateway capabilities have the following components:

- A configuration – A JSON document that defines all of the data sources for a capability.
- A namespace – A unique string that identifies the type and version of a capability. For example, the OPC-UA source capability namespace is `iotsitewise:opcuacollector:`*`version`*, where *`version`* is the version of the OPC-UA capability. All OPC-UA sources are defined in one capability with this namespace.
- A synchronization status – A status that indicates if a capability is synchronized between the AWS Cloud and the SiteWise Edge gateway. The sync status can be one of the following:
  - `IN_SYNC` – The SiteWise Edge gateway is running the capability configuration.
  - `OUT_OF_SYNC` – The SiteWise Edge gateway hasn't received the capability configuration.
  - `SYNC_FAILED` – The SiteWise Edge gateway rejected the capability configuration.

  After you update a capability configuration, its sync status is `OUT_OF_SYNC` until the SiteWise Edge gateway receives and applies or rejects the updated configuration.

Use the following operations to query and update your SiteWise Edge gateway sources and capability configurations:

- DescribeGateway – Retrieves information about a specific SiteWise Edge gateway. The response includes a list of capability summaries, including capability namespaces.
- DescribeGatewayCapabilityConfiguration – Retrieves the configuration of a specific capability. Use this operation to retrieve a capability configuration to update.
- ListGateways – Lists information about all SiteWise Edge gateways. The response includes a list of capability summaries for each SiteWise Edge gateway, including capability namespaces.

- UpdateGatewayCapabilityConfiguration – Updates a SiteWise Edge gateway capability configuration or defines a new capability configuration. This operation identifies capabilities by a capability namespace. If you provide a namespace that already exists, this operation updates the capability for that namespace. Otherwise, this operation creates a new capability.

  > **Warning**
  >
  > The UpdateGatewayCapabilityConfiguration operation overwrites the existing capability configuration with the configuration that you provide in the payload. To avoid deleting your capability's configuration, you must add to the existing configuration when you update the capability.

SiteWise Edge gateway capabilities

- ???
- ???
- ???

Choosing a destination for your source server data

Data is exported from the edge to AWS IoT SiteWise in real time, or in batches using Amazon S3. You can also send the stream to another component using a AWS IoT Greengrass stream.

- **AWS IoT SiteWise real-time** – Choose this to send data directly to AWS IoT SiteWise storage. Ingest and monitor data in real-time, and process data at the edge.
- **AWS IoT SiteWise Buffered using Amazon S3** – Send data in parquet format to Amazon S3 and then import into AWS IoT SiteWise storage. Choose this option to ingest data in batches, and store historical data in a cost-effective way. You can configure your preferred Amazon S3 bucket location, and the frequency at which you want data to be uploaded to Amazon S3. You can also choose what to do with the data after ingestion into AWS IoT SiteWise. You can choose to have the data available in both SiteWise and Amazon S3 or you can choose to delete it automatically from Amazon S3.
  - The Amazon S3 bucket is a staging and buffering mechanism and supports files in the parquet format.

- If you select the check box **Import data into AWS IoT SiteWise storage**, data is uploaded into Amazon S3 first, and then into AWS IoT SiteWise storage.
  - If you select the check box **Delete data from Amazon S3**, data is deleted from Amazon S3, after it is imported into SiteWise storage.
  - If you clear the check box **Delete data from Amazon S3**, data is stored both in Amazon S3, and in SiteWise storage.
- If you clear the check box **Import data into AWS IoT SiteWise storage**, data is stored only in Amazon S3. It is not imported into SiteWise storage.

  Visit Managing data storage for details on the various storage options AWS IoT SiteWise provides. To learn more about pricing options, see AWS IoT SiteWise pricing.
- **AWS IoT Greengrass stream manager** – Use AWS IoT Greengrass stream manager to send data to the following AWS Cloud destinations: channels in AWS IoT Analytics, streams in Amazon Kinesis Data Streams, asset properties in AWS IoT SiteWise, or objects in Amazon Simple Storage Service (Amazon S3). For more information, see Manage data streams on the AWS IoT Greengrass Core in *AWS IoT Greengrass Version 2 Developer Guide*.

The following example shows the required data stream message structure. All fields are required.

```
{
    "assetId": "string",
    "propertyAlias": "string",
    "propertyId": "string",
    "propertyValues": [
      {
         "quality": "string",
         "timestamp": {
            "offsetInNanos": number,
            "timeInSeconds": number
         },
         "value": {
            "booleanValue": boolean,
            "doubleValue": number,
            "integerValue": number,
            "stringValue": "string"
         }
      }
    ]
```

```
}
```

> **Note**
>
> The data stream message must include either (`assetId` and `propertyId`) or `propertyAlias` in its structure.

`assetId`

(Optional) The ID of the asset to update.

`propertyAlias`

(Optional) The alias that identifies the property, such as an OPC-UA server data stream path. For example:

```
/company/windfarm/3/turbine/7/temperature
```

For more information, see Mapping industrial data streams to asset properties in the *AWS IoT SiteWise User Guide*.

`propertyId`

(Optional) The ID of the asset property for this entry.

`propertyValues`

(Required) The list of property values to upload. You can specify up to 10 `propertyValues` array elements.

`quality`

(Optional) The quality of the asset property value.

`timestamp`

(Required) The timestamp of the asset property value.

`offsetInNanos`

(Optional) The nanosecond offset from `timeInSeconds`.

`timeInSeconds`

(Required) The timestamp date, in seconds, in the Unix epoch format. Fractional nanosecond data is provided by `offsetInNanos`.

`value`

(Required) The value of the asset property.

> **Note**
>
> Only one of the following values can exist in the `value` field.

`booleanValue`

(Optional) Asset property data of type Boolean (`true` or `false`).

`doubleValue`

(Optional) Asset property data of type double (floating point number).

`integerValue`

(Optional) Asset property data of type integer (whole number).

`stringValue`

(Optional) Asset property data of type string (sequence of characters).

Adding partner data sources to SiteWise Edge gateways

When using an AWS IoT SiteWise Edge gateway you can connect a partner data source to your SiteWise Edge gateway and receive data from the partner in your SiteWise Edge gateway and the AWS cloud. These partner data sources are AWS IoT Greengrass components that are developed in partnership between AWS and the partner. When you add a partner data source, AWS IoT SiteWise will create this component and deploy it on your SiteWise Edge gateway.

To add a partner data source, do the following:

- Add a partner data source

- Go to the partner's web portal and configure the partner data source so it connects to the SiteWise Edge gateway.

Topics

Security

As part of the Shared Responsibility Model between AWS, our customers, and our partners the following describes who is responsible for the different aspects of security:

Customer responsibility

- Vetting the partner.
- Configuring the network access given to the partner.

AWS responsibility

- Isolating the partner from the customer AWS cloud resources except those needed by the partner. In this case, AWS IoT SiteWise ingestion.
- Restricting the partner solution to a reasonable usage of the SiteWise Edge gateway machine resources (CPU, memory, file system).

Partner responsibility

- Using secure defaults.
- Keeping the solution secure over time through patches and other appropriate updates.
- Keeping customer data confidential.

Add a partner data source

To connect a partner data source to your SiteWise Edge gateway, add it as a data source. When you add it as a data source, AWS IoT SiteWise will deploy a private AWS IoT Greengrass component to your SiteWise Edge gateway.

Prerequisites

To add a partner data source, you must do the following:

- Create an account with the partner.
- Bind the accounts.

To create a SiteWise Edge gateway with a partner data source

If you want to create a new SiteWise Edge gateway, complete the steps in Creating a SiteWise Edge gateway. After you've created SiteWise Edge gateway follow the steps in To add a partner data source to an existing SiteWise Edge gateway to add a partner data source.

To add a partner data source to an existing SiteWise Edge gateway

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Gateways**.
3. Choose the SiteWise Edge gateway you want to connect the partner data source to.
4. Under **Data sources**, choose **Add data source**.
5. For **Source type**, choose the partner you want to connect your SiteWise Edge gateway to.

   > **Note**
   >
   > Currently, EasyEdge is the only available partner data source. The first time you add an EasyEdge data source, you'll need to create an EasyEdge account.

6. Enter a name for the source.
7. To grant the partner access to the data source, select **Authorize**.
8. To let AWS IoT SiteWise update your AWS IoT SiteWise publisher component and, if the data processing pack is enabled, the AWS IoT SiteWise processor component, select **Update components**.
9. Choose **Save**.

Set up docker on your SiteWise Edge gateway

To add a partner data source, Docker Engine 1.9.1 or later must be installed on your local device.

> **Note**
>
> Version 20.10 is the latest version that is verified to work with the SiteWise Edge gateway software.

To verify Docker is installed

To verify Docker is installed, run the following command from a terminal connected to your SiteWise Edge gateway:

```
docker info
```

If the command returns a `docker is not recognized` result, or an older version of Docker is installed, Install Docker Engine before continuing.

To set up Docker

The system user that runs a Docker container component must have root or administrator permissions, or you must configure Docker to run it as a non-root or non-admistrator user.

On Linux devices, you must add a `ggc_user` user to the `docker` group to call Docker commands without `sudo`.

To add `ggc_user`, or the non-root user that you use to run Docker container components, to the `docker` group, run the following command:

```
sudo usermod -aG docker ggc_user
```

For more information, see Linux post-installation steps for Docker Engine.

SiteWise Edge gateway partner data sources

Use the information below to configure a partner data source.

EasyEdge

Portal:

https://studio.easyedge.io/

EasyEdge documentation:

EasyEdge for AWS

Using packs

AWS IoT SiteWise Edge gateways use different packs to determine how to collect and process your data.

Currently, the following packs are available:

- **Data collection pack** – Use this pack to collect your industrial data and route it to AWS Cloud destinations. By default, this pack is enabled automatically for your SiteWise Edge gateway.
- **Data processing pack** – Use this pack to enable SiteWise Edge gateway communication with edge-configured asset models and assets. You can use edge configuration to control what asset data to compute and process on-site. You can then send your data to AWS IoT SiteWise or other AWS services. For more information about the data processing pack, see the section called "Enabling edge data processing".

Upgrading packs

> **Important**
>
> Upgrading Data processing pack versions from before (and including) 2.0.x to version 2.1.x will result in data loss of locally stored measurements.

SiteWise Edge gateways use different packs to determine how to collect and process your data. You can use the AWS IoT SiteWise console to upgrade packs.

To upgrade packs (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Gateways**.
3. In the **Gateways** list, choose the SiteWise Edge gateway with the packs you want to upgrade.
4. In the **Gateway configuration** section, choose **Software updates available**.

5. On the edit software versions page, in the **Gateway component updates** section, do the following:
   - To update the **OPC-UA collector**, choose a version, and then choose **Deploy**.
   - To update the **Publisher**, choose a version, and then choose **Deploy**.
   - To update the **Data processing pack**, choose a version, and then choose **Deploy**.
6. When you're done deploying new versions, choose **Done**.

If you're experiencing problems upgrading the packs, see Unable to deploy packs to SiteWise Edge gateways.

Managing SiteWise Edge gateways

You can use the AWS IoT SiteWise console and API operations to manage AWS IoT SiteWise Edge gateways. You can also use the AWS OpsHub for AWS IoT SiteWise for Windows application to manage some aspects of your SiteWise Edge gateway from your local device.

We highly recommend that you use the AWS OpsHub for AWS IoT SiteWise application to monitor the disk usage on your local device. You can also monitor the `Gateway.AvailableDiskSpace` and `Gateway.UsedPercentageDiskSpace` Amazon CloudWatch metrics and create alarms to get notified when the disk space is getting low. For more information about Amazon CloudWatch alarms, see Create a CloudWatch alarm based on a static threshold.

Make sure that your device has enough space for upcoming data. When you're about to run out of space on your local device, the service automatically deletes a small amount of data with the oldest timestamps to make room for upcoming data.

To check if the service deleted your data, do the following:

1. Sign in to the AWS OpsHub for AWS IoT SiteWise application.
2. Choose **Settings**.
3. For **Logs**, specify a time range, and then choose **Download**.
4. Unzip the log file.
5. If the log file contains the following message, the service deleted your data: *`number`* bytes of data have been deleted to prevent SiteWise Edge gateway storage from running out of space.

Managing your SiteWise Edge gateway with the AWS IoT SiteWise console

You can use the AWS IoT SiteWise console to configure, update, and monitor all SiteWise Edge gateways in your AWS account.

You can view your SiteWise Edge gateways by navigating to the **Edge Gateways** page in the AWS IoT SiteWise console. To access the **Edge gateway details** page for a specific gateway, choose the name of an Edge gateway.

From the **Overview** tab of the **Edge gateway details** page, you can do the following:

- In the **Data sources** section, update data source configuration and configure additional data sources
- Choose **Open CloudWatch metrics** to view the number of data points ingested per data source in the CloudWatch metrics console
- In the **Edge capabilities** section, add data packs to your SiteWise Edge gateway by clicking **Edit**
- In the **Gateway configuration** section, view the connectivity status of your SiteWise Edge gateways
- In the **Publisher configuration** section, view the SiteWise Edge gateway sync status and configuration of the AWS IoT SiteWise publisher component

From the **Updates** tab of the **Edge gateway details** page, you can see the current component and pack versions that are deployed to the Edge gateway. This is also where you deploy new versions, when they're available.

Managing SiteWise Edge gateways using AWS OpsHub for AWS IoT SiteWise

You use the AWS OpsHub for AWS IoT SiteWise application to manage and monitor your SiteWise Edge gateways. This application provides the following monitoring and management options:

- Under **Overview**, you can do the following:
  - View SiteWise Edge gateway details that help you get insights into your SiteWise Edge gateway device data, identify issues, and improve the SiteWise Edge gateway's performance.

  - View SiteWise Monitor portals that monitor the data from local servers and equipment at the edge. For more information, see What is AWS IoT SiteWise Monitor in the *AWS IoT SiteWise Monitor Application Guide*.
- Under **Health**, there's a dashboard that displays data from your SiteWise Edge gateway. Domain experts, such as process engineers, can use the dashboard to see an overview of SiteWise Edge gateway behavior.
- Under **Assets**, view assets deployed to the local device and the last value collected or computed for asset properties.
- Under **Settings**, you can do the following:
  - If the Data Processing Pack is installed, view the SiteWise Edge gateway configuration information and sync resources with the AWS Cloud.
  - Download the authentication files that you can use to access the SiteWise Edge gateway by using other tools.
  - Download logs that you can use to troubleshoot the SiteWise Edge gateway.
  - View the AWS IoT SiteWise components deployed to the SiteWise Edge gateway.

> **Important**
>
> The following are required to use AWS OpsHub for AWS IoT SiteWise:
>
> - Your local device and the AWS OpsHub for AWS IoT SiteWise application must be connected to the same network.
> - The data processing pack must be enabled.

To manage SiteWise Edge gateways using AWS OpsHub

1. Download and install the AWS OpsHub for AWS IoT SiteWise for Windows application.
2. Open the application.
3. If you don't have local credentials set up for your gateway, follow the steps under Accessing your SiteWise Edge gateway using local operating system credentials to set them up.
4. You can sign in to your SiteWise Edge gateway with your Linux or Lightweight Directory Access Protocol (LDAP) credentials. To sign in to your SiteWise Edge gateway, do one of the following:

Linux

1. For **Hostname or IP address**, enter the hostname or IP address of your local device.
2. For **Authentication**, choose **Linux**.
3. For **User name**, enter the user name of your Linux operating system.
4. For **Password**, enter the password of your Linux operating system.
5. Choose **Sign in**.

LDAP

1. For **Hostname or IP address**, enter the hostname or IP address of your local device.
2. For **Authentication**, choose **LDAP**.
3. For **User name**, enter your LDAP's user name.
4. For **Password**, enter your LDAP's password.
5. Choose **Sign in**.

Accessing your SiteWise Edge gateway using local operating system credentials

Besides Lightweight Directory Access Protocol (LDAP), you can use the Linux or Windows credentials to access your SiteWise Edge gateway.

> **Important**
>
> To access your SiteWise Edge gateway with Linux credentials, you must activate the data processing pack for your SiteWise Edge gateway.

Accessing your SiteWise Edge gateway using Linux operating system credentials

The following steps assume that you use a device with Ubuntu. If you use a different Linux distribution, consult the relevant documentation for your device.

To create a Linux user pool

1. To create an admin group, run the following command.

```
sudo groupadd --system SWE_ADMIN_GROUP
```

Users in the SWE_ADMIN_GROUP group can allow admin access for the SiteWise Edge gateway.

2. To create a user group, run the following command.

```
sudo groupadd --system SWE_USER_GROUP
```

Users in the SWE_USER_GROUP group can allow read-only access for the SiteWise Edge gateway.

3. To add a user to the admin group, run the following command. Replace *user-name* and *password* with the user name and password that you want to add.

```
sudo useradd -p $(openssl passwd -1 password) user-name
```

4. To add a user to either SWE_ADMIN_GROUP or SWE_USER_GROUP, replace *user-name* with the the user name that you added in the previous step.

```
sudo usermod -a -G SWE_ADMIN_GROUP user-name
```

You can now use the user name and password to sign in to the SiteWise Edge gateway on the AWS OpsHub for AWS IoT SiteWise application.

Accessing your SiteWise Edge gateway using Windows credentials

The following steps assume that you use a device with Windows.

> **Important**
>
> Security is a shared responsibility between AWS and you. Create a strong password policy with at least 12 characters and a combination of uppercase, lowercase, numbers, and symbols. Additionally, set the Windows Firewall rules to allow incoming traffic on port 443 and to block incoming traffic on all other ports.

To create a Windows Server user pool

1. Run PowerShell as the administrator.

a. On the Windows server where you want to install SiteWise Edge Gateway, log in as administrator.

b. Enter **PowerShell** in the Windows search bar.

c. In the search results, right click on the Windows PowerShell app. Choose **Run as Administrator**.

2. To create an admin group, run the following command.

```
net localgroup SWE_ADMIN_GROUP /add
```

You must be a user in the SWE_ADMIN_GROUP group to allow admin access for the SiteWise Edge gateway.

3. To create a user group, run the following command.

```
net localgroup SWE_USER_GROUP /add
```

You must be a user in the SWE_USER_GROUP group to allow ready-only access for the SiteWise Edge gateway.

4. To add user, run the following command. Replace *user-name* and *password* with the user name and the password that you want to create.

```
net user user-name password /add
```

5. To add a user to the admin group, run the following command. Replace *user-name* with the user name that you want to add.

```
net localgroup SWE_ADMIN_GROUP user-name /add
```

You can now use the user name and password to sign in to the SiteWise Edge gateway on the AWS OpsHub for AWS IoT SiteWise application.

Managing the SiteWise Edge gateway certificate

You can use SiteWise Monitor and third-party applications, such as Grafana, on your SiteWise Edge gateway devices. These applications require a TLS connection to the service. SiteWise Edge

gateways currently use a self-signed certificate. If you use a browser to open the applications, such as a SiteWise Monitor portal, you might receive a warning for untrusted certificate.

The following shows how to download the trusted certificate from the AWS OpsHub for AWS IoT SiteWise application.

1. Sign in to the application.
2. Choose **Settings**.
3. For **Authentication**, choose **Download certificate**.

The following assumes that you use Google Chrome or FireFox. If you use a different browser, consult the relevant documentation for your browser. To add the certificate that you downloaded in the previous step to a browser, do one of the following:

- If you use Google Chrome, follow the Set up certificates in the *Google Chrome Enterprise Help documentation*.
- If you use Firefox, follow the To Load the Certificate into the Mozilla or Firefox Browser in the *Oracle documentation*.

Changing the version of SiteWise Edge gateway component packs

You can use the AWS IoT SiteWise console to change the version of component packs on your SiteWise Edge gateways.

To change the version of a SiteWise Edge gateway component pack

1. Navigate to the AWS IoT SiteWise console.
2. In the left navigation pane, choose **Gateways**.
3. Select the SiteWise Edge gateway that you would like to change the pack versions for.
4. Under **Gateway configuration**, choose **View software versions**.
5. On the **Edit software versions** page, for the pack you want to update the version of, select the version you want to deploy and choose **Deploy**.
6. Choose **Done**.

Running SiteWise Edge on Siemens Industrial Edge

You can ingest data from your Siemens Industrial Edge device to your AWS account by running a SiteWise Edge gateway on the device. To do this, you create a SiteWise Edge gateway resource with a deployment target of **Siemens Industrial Edge device - new**, download the configuration file, and upload it to your Siemens app through the Siemens Industrial Edge Management (IEM) portal. For more information about running AWS IoT SiteWise Edge on Siemens Industrial Edge, including how to set up the required Siemens resources, see What is Industrial Edge? in the Siemens documentation.

> **Note**
>
> Siemens is not a vendor or supplier for AWS IoT SiteWise Edge. The Siemens Industrial Edge Marketplace is an independent marketplace.

Topics

Prerequisites

To run AWS IoT SiteWise Edge on Siemens Industrial Edge, you need the following:

- A Siemens Digital Exchange Platform account
- A Siemens Industrial Edge Hub (iehub) account
- A Siemens Industrial Edge Management (IEM) instance
- Either a Siemens Industrial Edge Device (IED) or a Siemens Industrial Edge virtual Device (IEvD)
- Access to the **Siemens Industrial Edge device** deployment target. To get access, go to the AWS IoT SiteWise console and choose **Request access**.

Security

As part of the Shared Responsibility Model between AWS, our customers, and our partners the following describes who is responsible for the different aspects of security:

Customer responsibility

- Vetting the partner.
- Configuring the network access given to the partner.
- Physically securing the device running AWS IoT SiteWise Edge.

AWS responsibility

- Isolating the partner from the customer AWS cloud resources.

Partner responsibility

- Using secure defaults.
- Keeping the solution secure over time through patches and other appropriate updates.
- Keeping customer data confidential.
- Vetting other applications available in the partner marketplace.

During the preview stage of this feature, customer data that AWS IoT SiteWise caches on the partner device is accessible by the partner and other applications installed through the partner marketplace.

Creating the configuration file

Once you have the proper Siemens accounts and IEM instances, you can create a SiteWise Edge gateway of deployment type **Siemens Industrial Edge device**.

To create the configuration file

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Edge gateways**.
3. Choose **Create gateway**.
4. For **Deployment type**, choose **Siemens Industrial Edge device - new**.
5. Enter a name for your SiteWise Edge gateway or use the name generated by AWS IoT SiteWise.
6. (Optional) Under **advanced configuration**, do the following:

- Enter a name for your AWS IoT Core Thing or use the name generated by AWS IoT SiteWise.

7. Choose **Create gateway**.
8. In the **Generate SiteWise Edge gateway configuration file** dialog box, choose **Generate and download**. AWS IoT SiteWise automatically generates a configuration file that you will use to configure the AWS IoT SiteWise Edge application.

> **Important**
>
> Make sure that you save the configuration file in a secure location. You will use the file later.

Now that you've created the SiteWise Edge gateway, do the following to finish setting up your SiteWise Edge gateway:

1. Add data sources
2. Configure the Publisher component

Once you have the configuration file and the SiteWise Edge gateway is configured, download the AWS IoT SiteWise Edge application from the Siemens Industrial Edge Marketplace and install it using the Siemens Industrial Edge Management (IEM) portal. Then, access your Siemens Industrial Edge device through the Siemens Industrial Edge Management (IEM) portal and upload the configuration file on the device where you want to install the SiteWise Edge gateway.

Troubleshooting

To troubleshoot the SiteWise Edge gateway on your Siemens Industrial Edge device, you can access the logs for the application through the Siemens Industrial Edge Management (IEM) or Siemens Industrial Edge Device (IED) portals. For more information, see Downloading Logs in the Siemens documentation.

I see 'SESSION_TAKEN_OVER' or 'com.aws.greengrass.mqttclient.MqttClient: Failed to publish the message via Spooler and will retry.' in the logs

If you see a warning that includes `SESSION_TAKEN_OVER` or an error that includes `com.aws.greengrass.mqttclient.MqttClient: Failed to publish the message via`

`Spooler and will retry.` in your logs at `/greengrass/v2/logs/greengrass.log`, you may be trying to use the same configuration file for multiple SiteWise Edge gateways on multiple devices. Each SiteWise Edge gateway needs a unique configuration file to connect to your AWS account.

I see 'com.aws.greengrass.deployment.IotJobsHelper: No deployment job found.' or 'Deployment result already reported.' in the logs

If you see `com.aws.greengrass.deployment.IotJobsHelper: No deployment job found.` or `Deployment result already reported.` in your logs at `/greengrass/v2/logs/greengrass.log`, you may be trying to reuse the same configuration file.

There are multiple solutions:

- If you want to reuse the configuration file, do the following:
  1. Navigate to the AWS IoT SiteWise console.
  2. In the navigation pane, choose **Gateways**.
  3. Choose the SiteWise Edge gateway you want to reuse.
  4. Choose the **Updates** tab.
  5. Select a different Publisher version and choose **Deploy**.
- Follow the steps in Creating the configuration file to create a new configuration file.

I see 'Config file missing AWS_REGION' in the logs.

If you see `Config file missing AWS_REGION` in the Siemens logs, the JSON of the configuration file has been corrupted. You'll need to create a new configuration file. Follow the steps in Creating the configuration file to create a new configuration file.

Contact us

- If you'd like to request access to the application, go to the AWS IoT SiteWise console and choose **Request access**.
- If you'd like help troubleshooting the application, go to the AWS IoT SiteWise console, navigate to the details page of the SiteWise Edge gateway, and choose **Get support**.

Filtering assets on a SiteWise Edge gateway

You can use edge filtering to more efficiently manage your assets by sending only a subset of assets to a specific SiteWise Edge gateway for use in data processing. If your assets are arranged in a tree, or parent-child, structure, you can set up an IAM policy attached to a SiteWise Edge gateway's IAM role that only allows the root of the tree, or parent, and its children to be sent to a specific SiteWise Edge gateway.

> **Note**
>
> If you're arranging existing assets into a tree structure, after you've created the structure, go into each existing asset that you added to the structure and choose **Edit** and then choose **Save** to make sure AWS IoT SiteWise recognizes the new structure.

Setting up edge filtering

Set up edge filtering on your SiteWise Edge gateway by adding the following IAM policy to the SiteWise Edge gateway's IAM role, replacing *<root-asset-id>* with the ID of the root asset you want to send to the SiteWise Edge gateway.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iotsitewise:DescribeAsset",
                "iotsitewise:ListAssociatedAssets"
            ],
            "Resource": "arn:aws:iotsitewise:*:*:asset/*",
            "Condition": {
                "StringNotLike": {
                    "iotsitewise:assetHierarchyPath": "/<root-asset-id>*"
                }
            }
        }
    ]
}
```

If there are assets currently on your SiteWise Edge gateway that you'd like to remove, log into your SiteWise Edge gateway and run the following command to force the SiteWise Edge gateway to sync with AWS IoT SiteWise by deleting the cache.

```
sudo rm /greengrass/v2/work/aws.iot.SiteWiseEdgeProcessor/sync-app/
sync_resource_bundles/edge.json
```

Using AWS IoT SiteWise APIs on the edge

You can use a subset of the available AWS IoT SiteWise APIs along with edge-specific APIs to interact with asset models and their assets on the edge. The asset models must be configured to run at the edge. For more information, see Processing data at the edge.

Use these APIs to gather data about your asset models and assets, monitor your deployed portals and dashboard metrics, and get asset data gathered at the edge. This provides a central host in your network for interactions with AWS IoT SiteWise without requiring a web API call.

Topics

- All available APIs for use with AWS IoT SiteWise edge devices
- Edge-only APIs for use with AWS IoT SiteWise edge devices
- Tutorial: Getting a list of asset models on a SiteWise Edge gateway

All available APIs for use with AWS IoT SiteWise edge devices

When working with devices on the edge you can use a variety of APIs to interact with AWS IoT SiteWise and complete tasks locally on the device.

Available AWS IoT SiteWise APIs

The following AWS IoT SiteWise APIs are available on edge devices:

- ListAssetModels
- DescribeAssetModel
- ListAssets
- DescribeAsset
- DescribeAssetProperty

- ListAssociatedAssets
- GetAssetPropertyAggregates
- GetAssetPropertyValue
- GetAssetPropertyValueHistory
- ListDashboards
- ListPortals
- ListProjectAssets
- ListProjects
- DescribeDashboard
- DescribePortal
- DescribeProject

Available edge-only APIs

The following APIs are used locally on devices on the edge:

- Authenticate – Use this API to get the SigV4 temporary credentials that you'll use to make API calls.

Edge-only APIs for use with AWS IoT SiteWise edge devices

In addition to the AWS IoT SiteWise APIs that are available on the edge, there are edge-specific ones. Those edge-specifc APIs are described below.

Authenticate

Gets the credentials from the SiteWise Edge gateway. You'll need to add local users or connect to your system using LDAP or a Linux user pool. For more information about adding users, see LDAP or Linux user pool.

Request syntax

```
POST /authenticate HTTP/1.1
Content-type: application/json
{
  "username": "string",
  "password": "string",
```

```
  "authMechanism": "string"
}
```

URI request Parameters

The request does not use any URI parameters.

Request body

The request accepts the following data in JSON format.

username

The username used to validate the request call.

Type: String

Required: Yes

password

The password of the user requesting credentials.

Type: String

Required: Yes

authMechanism

The authentication method to validate this user in the host.

Type: String

Valid values: `ldap`, `linux`, `winnt`

Required: Yes

Response syntax

```
HTTP/1.1 200
Content-type: application/json
{
  "accessKeyId": "string",
  "secretAccessKey": "string",
  "sessionToken": "string",
  "region": "edge"
```

```
}
```

Response elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format.

accessKeyId

The access key ID that identifies the temporary security credentials.

Length Constraints: Minimum length of 16. Maximum length of 128.

Pattern: `[\w]*`

secretAccessKey

The secret access key that can be used to sign requests.

Type: String

sessionToken

The token that users must pass to the service API to use the temporary credentials.

Type: String

region

The region you are targeting for API calls.

Type: CONSTANT - `edge`

Errors

IllegalArgumentException

The request was rejected because the provided body document was malformed. The error message describes the specific error.

HTTP Status Code: 400

AccessDeniedException

The user doesn't have valid credentials based on the current Identity Provider. The error message describes the authentication Mechanism.

HTTP Status Code: 403

TooManyRequestsException

The request has reached it's limit of authentication attempts. The error message contains the the quantity of time to wait until new attempts of authentication are made.

HTTP Status Code: 429

Tutorial: Getting a list of asset models on a SiteWise Edge gateway

You can use a subset of the available AWS IoT SiteWise APIs along with edge-specific APIs to interact with asset models and their assets on the edge. This tutorial will walk you through getting temporary credentials to a AWS IoT SiteWise Edge gateway and getting a list of the asset models on the SiteWise Edge gateway.

Prerequisites

In the steps of this tutorial you can use a variety of tools. To use these tools, make sure you have the corresponding prerequisites installed.

To complete this tutorial, you need the following:

- A deployed and running SiteWise Edge gateway requirements
- Access to your SiteWise Edge gateway in the same network over port 443.
- OpenSSL installed
- (AWS OpsHub for AWS IoT SiteWise) The AWS OpsHub for AWS IoT SiteWise application
- (curl) curl installed
- (Python) urllib3 installed
- (Python) Python3 installed
- (Python) Boto3 installed
- (Python) BotoCore installed

Step 1: Get a SiteWise Edge gateway service signed certificate

To establish a TLS connection to the APIs available at the SiteWise Edge gateway, you need a trusted certificate. You can generate this certificate using a OpenSSL or AWS OpsHub for AWS IoT SiteWise.

OpenSSL

> **Note**
>
> You need OpenSSL installed to run this command.

Open a terminal and run the following command to get a signed certificate from the SiteWise Edge gateway. Replace `<sitewise_gateway_ip>` with the IP of the SiteWise Edge gateway.

```
openssl s_client -connect <sitewise_gateway_ip>:443 </dev/null 2>/dev/null | openssl
 x509 -outform PEM > GatewayCert.pem
```

AWS OpsHub for AWS IoT SiteWise

You can use AWS OpsHub for AWS IoT SiteWise. For more information, see Managing SiteWise Edge gateways.

The absolute path to the downloaded SiteWise Edge gateway certificate is used in this tutorial. Run the following command to export the complete path of your certificate, replacing `<absolute_path_to_certificate>` with the path to the certificate:

```
export PATH_TO_CERTIFICATE='<absolute_path_to_certificate>'
```

Step 2: Get your SiteWise Edge gateway hostname

> **Note**
>
> You need OpenSSL installed to run this command.

To complete the tutorial you'll need the hostname of your SiteWise Edge gateway. To get the hostname of your SiteWise Edge gateway, run the following, replacing `<sitewise_gateway_ip>` with the IP of the SiteWise Edge gateway:

```
openssl s_client -connect <sitewise_gateway_ip>:443 </dev/null 2>/dev/null | grep -Po
 'CN = \K.*'| head -1
```

Run the following command to export the hostname for use later, replacing <your_edge_gateway_hostname> with the hostname of your SiteWise Edge gateway:

```
export GATEWAY_HOSTNAME='<your_edge_gateway_hostname>'
```

Step 3: Get temporary credentials for your SiteWise Edge gateway

Now that you have the signed certificate and the hostname of your SiteWise Edge gateway, you need to get temporary credentials so you can run APIs on the gateway. You can get these credentials through AWS OpsHub for AWS IoT SiteWise or directly from the SiteWise Edge gateway using APIs.

> **Important**
>
> Credentials expire every 4 hours, so you should get the credentials just before using the APIs on your SiteWise Edge gateway. Don't cache credentials for longer than 4 hours.

Get temporary credentials using AWS OpsHub for AWS IoT SiteWise

> **Note**
>
> You need the AWS OpsHub for AWS IoT SiteWise application installed.

To use AWS OpsHub for AWS IoT SiteWise application to get your temporary credentials do the following:

1. Log into the application.
2. Choose **Settings**.
3. For **Authentication**, choose **Copy credentials**.
4. Expand the option that fits your environment and choose **Copy**.
5. Save the credentials for use later.

Get temporary credentials using the SiteWise Edge gateway API

To use the SiteWise Edge gateway API to get the temporary credentials you can use a Python script or curl, first you'll need to have a user name and password for your SiteWise Edge gateway. The

SiteWise Edge gateways use SigV4 authentication and authorization. For more information about adding users, see LDAP or Linux user pool. These credentials will be used in the following steps to get the local credentials on your SiteWise Edge gateway that are needed to use the AWS IoT SiteWise APIs.

Python

> **Note**
>
> You need urllib3 and Python3 installed.

To get the credentials using Python

1. Create a file called **get_credentials.py** and the copy the following code into it.

```
'''
The following demonstrates how to get the credentials from the SiteWise Edge
 gateway. You will need to add local users or connect your system to LDAP/AD
https://docs.aws.amazon.com/iot-sitewise/latest/userguide/manage-gateways-
ggv2.html#create-user-pool

Example usage:
    python3 get_credentials.py -e https://<gateway_hostname> -c
 <path_to_certificate> -u '<gateway_username>' -p '<gateway_password>' -m
 '<method>'
'''
import urllib3
import json
import urllib.parse
import sys
import os
import getopt

"""
This function retrieves the AWS IoT SiteWise Edge gateway credentials.
"""
def get_credentials(endpoint,certificatePath, user, password, method):
    http = urllib3.PoolManager(cert_reqs='CERT_REQUIRED', ca_certs=
 certificatePath)
    encoded_body = json.dumps({
        "username": user,
```

```
        "password": password,
        "authMechanism": method,
    })

    url =  urllib.parse.urljoin(endpoint, "/authenticate")

    response = http.request('POST', url,
        headers={'Content-Type': 'application/json'},
        body=encoded_body)

    if response.status != 200:
        raise Exception(f'Failed to authenticate! Response status
 {response.status}')

    auth_data = json.loads(response.data.decode('utf-8'))

    accessKeyId = auth_data["accessKeyId"]
    secretAccessKey = auth_data["secretAccessKey"]
    sessionToken = auth_data["sessionToken"]
    region = "edge"

    return accessKeyId, secretAccessKey, sessionToken, region

def print_help():
    print('Usage:')
    print(f'{os.path.basename(__file__)} -e <endpoint> -c <path/to/certificate>
 -u <user> -p <password> -m <method> -a <alias>')
    print('')
    print('-e, --endpoint   edge gateway endpoint. Usually the Edge gateway
 hostname.')
    print('-c, --cert_path path to downloaded gateway certificate')
    print('-u, --user       Edge user')
    print('-p, --password   Edge password')
    print('-m, --method     (Optional) Authentication method (linux, winnt,
 ldap), default is linux')
    sys.exit()


def parse_args(argv):
    endpoint = ""
    certificatePath = None
    user = None
    password = None
    method = "linux"
```

```
    try:
        opts, args = getopt.getopt(argv, "he:c:u:p:m:",
 ["endpoint=","cert_path=", "user=", "password=", "method="])
    except getopt.GetoptError:
        print_help()

    for opt, arg in opts:
        if opt == '-h':
            print_help()
        elif opt in ("-e", "--endpoint"):
            endpoint = arg
        elif opt in ("-u", "--user"):
            user = arg
        elif opt in ("-p", "--password"):
            password = arg
        elif opt in ("-m", "--method"):
            method = arg.lower()
        elif opt in ("-c", "--cert_path"):
            certificatePath = arg

    if method not in ['ldap', 'linux', 'winnt']:
        print("not valid method parameter, required are ldap, linux, winnt")
        print_help()

    if (user == None or password == None):
        print("To authenticate against edge user, password have to be passed
 together, and the region has to be set to 'edge'")
        print_help()

    if(endpoint == ""):
        print("You must provide a valid and reachable gateway hostname")
        print_help()

    return endpoint,certificatePath, user, password, method


def main(argv):
    # get the command line args
    endpoint, certificatePath, user, password, method = parse_args(argv)

    accessKeyId, secretAccessKey, sessionToken, region=get_credentials(endpoint,
 certificatePath, user, password, method)
```

```
    print("Copy and paste the following credentials into the shell, they are
 valid for 4 hours:")
    print(f"export AWS_ACCESS_KEY_ID={accessKeyId}")
    print(f"export AWS_SECRET_ACCESS_KEY={secretAccessKey}")
    print(f"export AWS_SESSION_TOKEN={sessionToken}")
    print(f"export AWS_REGION={region}")
    print()




if __name__ == "__main__":
   main(sys.argv[1:])
```

2. Run **get_credentials.py** from the terminal replacing <gateway_username> and <gateway_password> with the credentials you created.

```
python3 get_credentials.py -e https://$GATEWAY_HOSTNAME -c $PATH_TO_CERTIFICATE
 -u '<gateway_username>' -p '<gateway_password>' -m 'linux'
```

curl

> **Note**
>
> You need curl installed.

To get the credentials using curl

1. Run the following command from the terminal replacing <gateway_username> and <gateway_password> with the credentials you created.

```
curl --cacert $PATH_TO_CERTIFICATE --location \
-X POST https://$GATEWAY_HOSTNAME:443/authenticate \
--header 'Content-Type: application/json' \
--data-raw '{
    "username": "<gateway_username>",
    "password": "<gateway_password>",
    "authMechanism": "linux"
}'
```

The response should look like the following:

```
{
    "username": "sweuser",
    "accessKeyId": "<accessKeyId>",
    "secretAccessKey": "<secretAccessKey>",
    "sessionToken": "<sessionToken>",
    "sessionExpiryTime": "2022-11-17T04:51:40.927095Z",
    "authMechanism": "linux",
    "role": "edge-user"
}
```

2. Run the following command from your terminal.

```
export AWS_ACCESS_KEY_ID=<accessKeyId>
export AWS_SECRET_ACCESS_KEY=<secretAccessKey>
export AWS_SESSION_TOKEN=<sessionToken>
export AWS_REGION=edge
```

Step 4: Get a list of the asset models on the SiteWise Edge gateway

Now that you have a signed certificate, your SiteWise Edge gateway hostname, and temporary credentials for your SiteWise Edge gateway, you can use the `ListAssetModels` API to get a list of the asset models on your SiteWise Edge gateway.

Python

> **Note**
>
> You need Python3, Boto3, and BotoCore installed.

To get the the list of asset models using Python

1. Create a file called **list_asset_model.py** and the copy the following code into it.

```
import json
import boto3
import botocore
import os
```

```
# create the client using the credentials
client = boto3.client("iotsitewise",
    endpoint_url= "https://"+ os.getenv("GATEWAY_HOSTNAME"),
    region_name=os.getenv("AWS_REGION"),
    aws_access_key_id=os.getenv("AWS_ACCESS_KEY_ID"),
    aws_secret_access_key=os.getenv("AWS_SECRET_ACCESS_KEY"),
    aws_session_token=os.getenv("AWS_SESSION_TOKEN"),
    verify=os.getenv("PATH_TO_CERTIFICATE"),
    config=botocore.config.Config(inject_host_prefix=False))

# call the api using local credentials
response = client.list_asset_models()
print(response)
```

2. Run **list_asset_model.py** from the terminal.

```
python3 list_asset_model.py
```

curl

> **Note**
>
> You need curl installed.

To get the list of asset models using curl

Run the following command from the terminal.

```
curl \
  --request GET https://$GATEWAY_HOSTNAME:443/asset-models \
  --cacert $PATH_TO_CERTIFICATE \
  --aws-sigv4 "aws:amz:edge:iotsitewise" \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  -H "x-amz-security-token:$AWS_SESSION_TOKEN"
```

The response should look like the following:

```
{
    "assetModelSummaries": [
```

```
        {
            "arn": "arn:aws:iotsitewise:{region}:{account-id}:asset-model/{asset-
model-id}",
            "creationDate": 1.669245291E9,
            "description": "This is a small example asset model",
            "id": "{asset-model-id}",
            "lastUpdateDate": 1.669249038E9,
            "name": "Some Metrics Model",
            "status": {
                "error": null,
                "state": "ACTIVE"
            }
        },
        .
        .
        .
    ],
    "nextToken": null
}
```

Backup and restore SiteWise Edge gateways

This topic covers how to restore SiteWise Edge gateways and backup your metric data. If you are experiencing issues with a broken SiteWise Edge gateway on the same machine and need to troubleshoot the issue, please read the AWS IoT SiteWise documentation Troubleshooting SiteWise Edge gateway issues.

> **Note**
>
> The guidance covered in this topic is for SiteWise Edge gateways installed on AWS IoT Greengrass V2 version 2.1.0 or higher.

Daily backups of metric data

Creating a back up is important, if you would like to transfer or restore the data on a new machine. Backing up your data greatly reduces the risk of loss of operating data during a transfer or restoration process.

The **influxdb** folder path is as follows:

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgeProcessor/influxdb
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgeProcessor\influxdb
```

We recommend that you backup the whole folder with everything underneath it.

We recommend that you periodically backup your metric data from the 1.0 SiteWise Edge to either an external hard drive or to the AWS cloud.

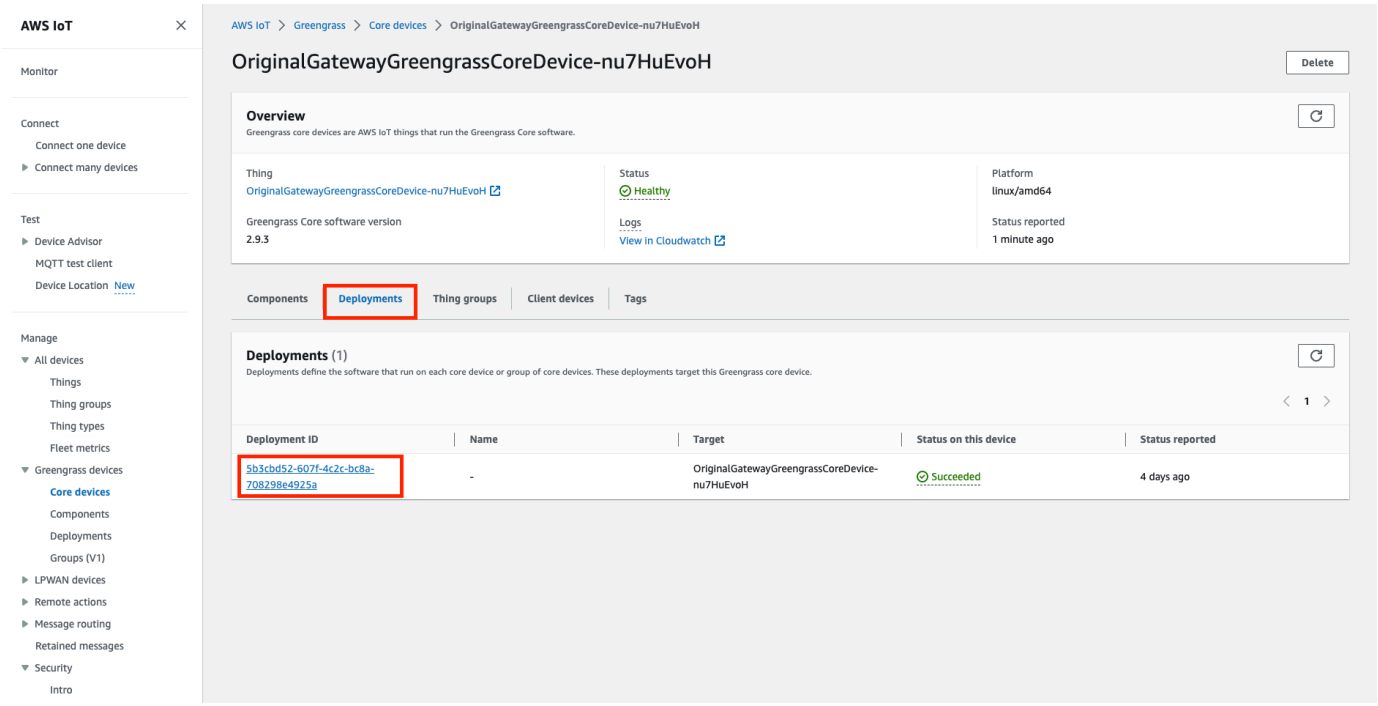
Restore a SiteWise Edge gateway

Use the following procedure to a restore a SiteWise Edge gateway:

1. Use the installation script downloaded when you create SiteWise Edge gateway to restore the SiteWise Edge gateway on the new machine. Read the Installing the SiteWise Edge gateway software on your local device procedure to setup the SiteWise Edge gateway.

   If you lose or cannot find the installation script, please contact AWS Customer Support.
2. Once the SiteWise Edge gateway has been installed, log into the AWS IoT Greengrass console.
3. To redeploy the components, navigate to **Manage** then under **AWS IoT Greengrass devices** select **Core devices**.
4. In the **AWS IoT Greengrass core devices** table select the core device corresponding to your SiteWise Edge gateway.

5. Once on the device page, open the **Deployments** tab and select your **Deployment ID**, this will open the **Deployments** page with your selected ID.



6. Once you are on the **Deployments** page, in the top right press the **Actions** button, and select the **Revise** option. to initiate a new deployment. Configure the deployment. If you would like to keep the deployment as it is, skip to **Review** and **Deploy**.

7. Wait for the **Deployment Status** to become `Completed`.

> **Note**
>
> It will also take a few minutes for all components on the SiteWise Edge to fully setup and running.

Restore AWS IoT SiteWise data

Use the following procedure to restore data on a new machine.

1. Copy the `influxdb` folder to the new machine.
2. Stop the SiteWise EdgeProcessor component, by running the following command in your terminal:

Linux

```
sudo /greengrass/v2/bin/greengrass-cli component stop -n
aws.iot.SiteWiseEdgeProcessor
```

Windows

```
C:\greengrass\v2\bin\greengrass-cli component stop -n
aws.iot.SiteWiseEdgeProcesso
```

3. Locate the path where you backed up your data, and run the following command:

   Linux

   ```
   sudo yes | sudo cp -rf <influxdb_backup_path> /greengrass/v2/work/
   aws.iot.SiteWiseEdgeProcessor/influxdb
   ```

   PowerShell

   ```
   Copy-Item -Recurse -Force <influxdb_backup_path>\* C:\greengrass
   \v2\work\aws.iot.SiteWiseEdgeProcessor\
   ```

   Windows

   ```
   robocopy <influxdb_backup_path> C:\greengrass\v2\work
   \aws.iot.SiteWiseEdgeProcessor\ /E
   ```

4. Restart the SiteWiseEdgeProcessor component:

   Linux

   ```
   sudo /greengrass/v2/bin/greengrass-cli component restart -n
   aws.iot.SiteWiseEdgeProcessor
   ```

   Windows

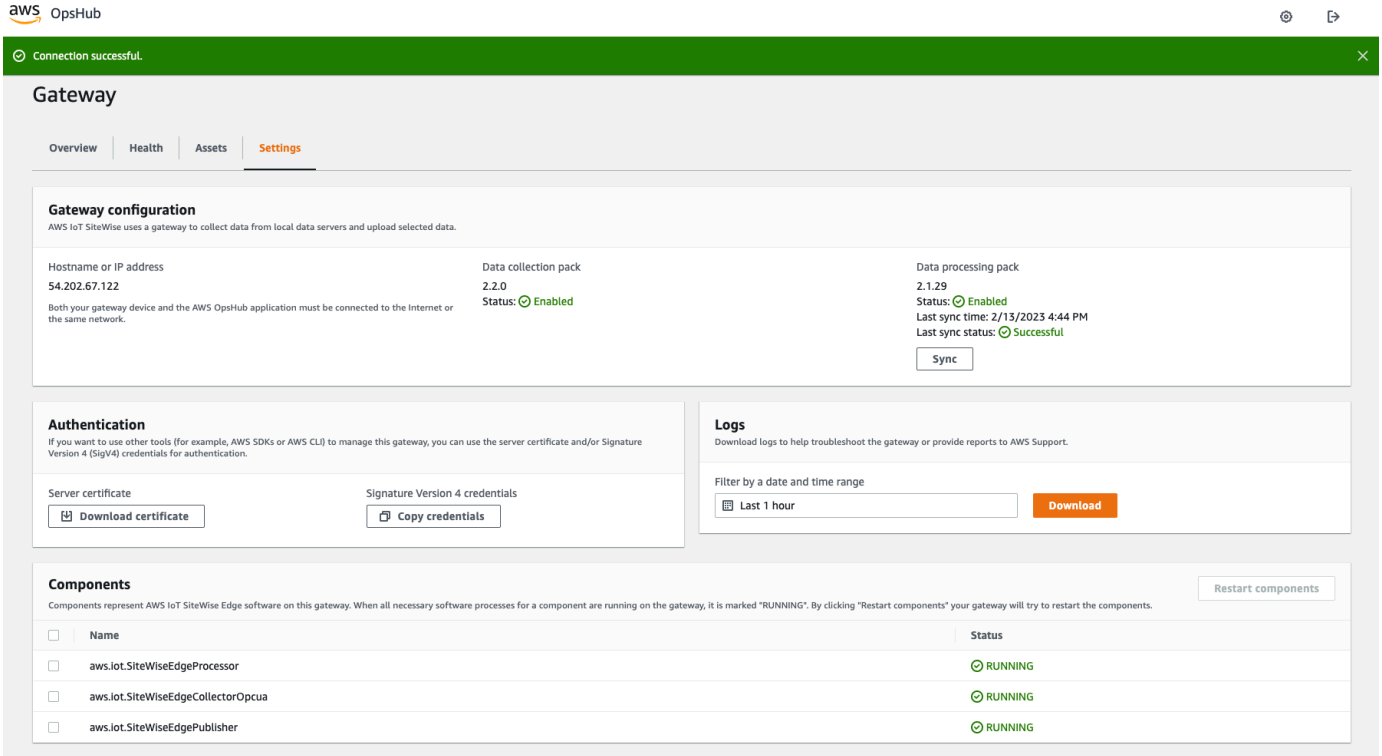
   ```
   C:\greengrass\v2\bin\greengrass-cli component restart -n
   aws.iot.SiteWiseEdgeProcessor
   ```

Validate successful backups and restorations

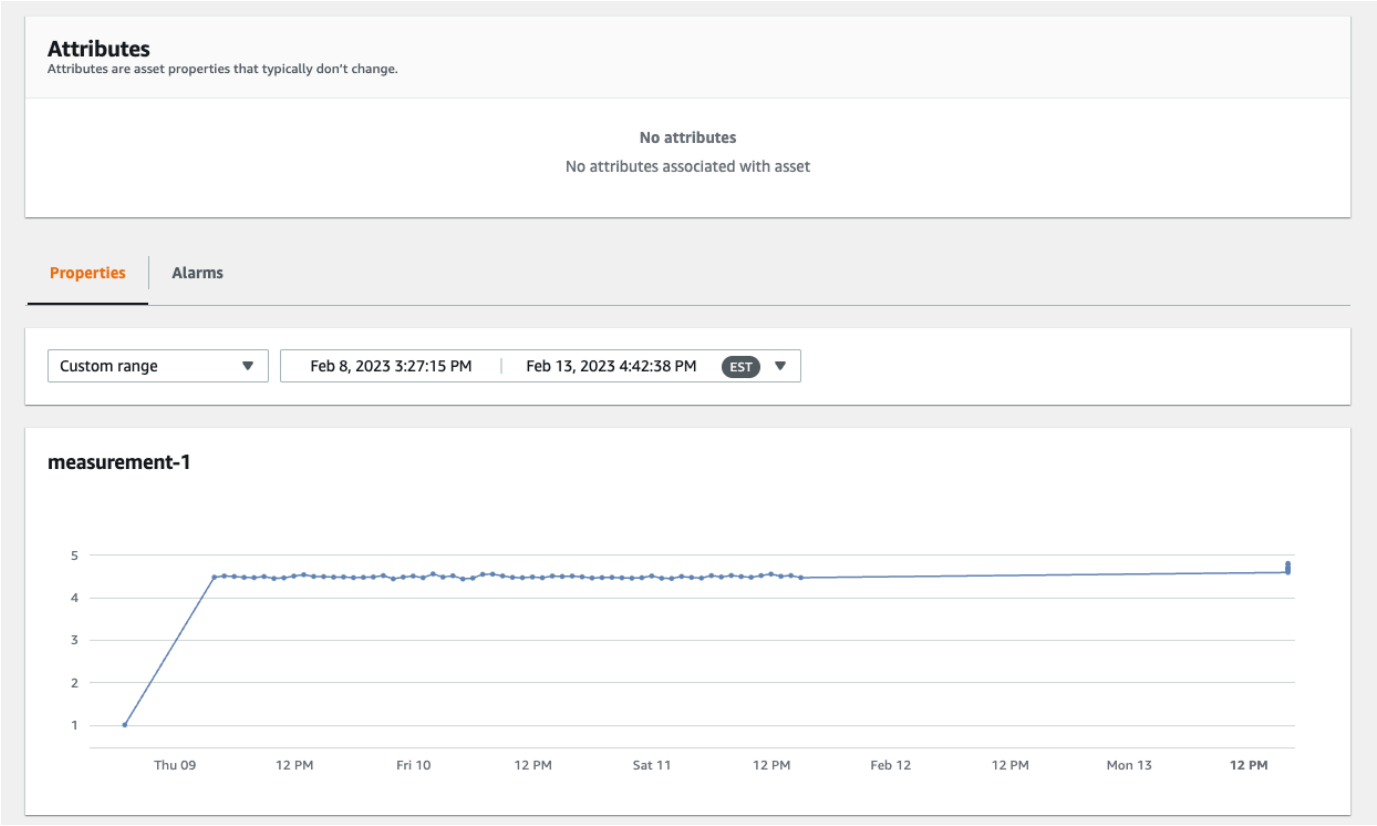
Use this procedure validate your backed-up data and SiteWise Edge gateway restorations.

> **Note**
>
> This procedure requires that you have installed AWS OpsHub for AWS IoT SiteWise. For more information see, Managing SiteWise Edge gateways using AWS OpsHub for AWS IoT SiteWise.

1. Open AWS OpsHub for AWS IoT SiteWise.
2. On the SiteWise Edge Gateway **Settings** page, check the status of each component listed in the **Components** table. Verify that the status color is green and the readout displays **RUNNING**.



3. Validate your past data on the portal dashboard to check that the past data and the new data are both properly setup. There will be a downtime between past and new data. You should except to see a duration where no data points are collected.



If you run into issues with backing up or restoring a SiteWise Edge gateway see the following troubleshooting topics [Troubleshooting an AWS IoT SiteWise Edge gateway](#).

Setting up SiteWise Edge gateways (AWS IoT Greengrass Version 1)

> **Note**
>
> SiteWise Edge gateways running on AWS IoT Greengrass V1 are available only if you started using this feature before July 29, 2021. Otherwise, you [set up SiteWise Edge gateways running on AWS IoT Greengrass V2](#).

You can send industrial data to AWS IoT SiteWise using an SiteWise Edge gateway to upload data from industrial equipment. The SiteWise Edge gateway serves as the intermediary between AWS IoT SiteWise and your data industrial equipment. AWS IoT SiteWise provides AWS IoT Greengrass

components that you can deploy on any device that can run AWS IoT Greengrass to set up a SiteWise Edge gateway. AWS IoT SiteWise supports linking with OPC-UA server protocol.

If you have AWS IoT SiteWise Edge gateways that run on AWS IoT Greengrass V1, you can upgrade your SiteWise Edge gateways to AWS IoT Greengrass V2. For more information, see Instructions for upgrading SiteWise Edge gateways from AWS IoT Greengrass V1 to AWS IoT Greengrass V2.

Topics

Choosing a AWS IoT Greengrass V1 SiteWise Edge gateway device

Choose a local device that best suits your industrial operation. You can configure a SiteWise Edge gateway on any device that can run AWS IoT Greengrass. All local devices must meet the following requirements:

- Supports AWS IoT Greengrass Core software v1.10.2 or later. For more information, see Supported platforms and requirements in the *AWS IoT Greengrass Version 1 Developer Guide*.
- Has at least 4 GB of RAM.
- Has at least 10 GB of free disk space.
- Supports a Java 8 virtual machine (JVM).

If you plan to process data at the edge with AWS IoT SiteWise, your local device must also meet the following requirements:

- Has an x86 64 bit quad-core processor.
- Has at least 16 GB of RAM.
- Has at least 32 GB for RAM if using Windows.
- Had at least 256 GB of free disk space.

The disk space required for caching data for intermittent internet connectivity depends on the following factors:

- Number of data streams uploaded

- Data points per data stream per second
- Size of each data point
- Communication speeds
- Expected network downtime

The compute capacity required to poll and upload data depends on the following factors:

- Number of data streams uploaded
- Data points per data stream per second

Configuring a AWS IoT Greengrass V1 SiteWise Edge gateway

A AWS IoT SiteWise Edge gateway serves as the intermediary between your industrial equipment and AWS IoT SiteWise. You can deploy the SiteWise Edge gateway software on any device that can run AWS IoT Greengrass. For more information, see Choosing a AWS IoT Greengrass V1 SiteWise Edge gateway device.

You can enable AWS IoT SiteWise to process data locally on your edge devices by using the data processing pack on your SiteWise Edge gateway. You do this when you add your SiteWise Edge gateway to AWS IoT SiteWise. For more information about processing data at the edge, see the section called “Enabling edge data processing”.

> **Note**
>
> We recommend that you complete the following steps with someone who has IT administrative access to your local and corporate networks. These steps might require someone with knowledge of your industrial equipment and the authority to configure firewall settings.

Topics

- Setting up the SiteWise Edge gateway environment
- Creating an IAM policy and role
- Configuring an AWS IoT Greengrass group
- Configuring the AWS IoT SiteWise connector

- Adding the SiteWise Edge gateway to AWS IoT SiteWise

Setting up the SiteWise Edge gateway environment

In this procedure, you install AWS IoT Greengrass and configure your SiteWise Edge gateway to use with AWS IoT SiteWise.

> **Note**
>
> This section includes instructions to install packages using the `apt` command. This is applicable to systems running Ubuntu or similar. If you aren't using a similar system, consult the documentation for your distribution and use the recommended package installer.

To set up the SiteWise Edge gateway

1. As appropriate, modify the BIOS settings of the SiteWise Edge gateway as follows.
   a. Ensure that the SiteWise Edge gateway automatically restarts after a potential power failure, if applicable.
   b. Ensure that the SiteWise Edge gateway won't hibernate or sleep, if applicable.
2. Ensure that the SiteWise Edge gateway connects to the internet.
3. (Optional) To use the SiteWise Edge gateway without the mouse, keyboard, and monitor, do the following steps to set up `ssh` on the SiteWise Edge gateway:
   a. If you haven't already installed the SSH package, run the following command.

      ```
      sudo apt install ssh
      ```

   b. Run the following command.

      ```
      service ssh status
      ```

   c. Search for `Active: active (running)` in the output to confirm that the SSH server is running,
   d. Press **Q** to exit.

Run the following command to use SSH to connect to the SiteWise Edge gateway from another computer. Replace *username* with the user login and *IP* with the IP address of the SiteWise Edge gateway.

```
ssh username@IP
```

You can use the `-p` *port-number* argument to connect to a port other than the default port 22.

4. Download and install AWS IoT Greengrass Core software v1.10.2 or later, and create an AWS IoT Greengrass group for your SiteWise Edge gateway. To do so, follow the instructions in Getting started with AWS IoT Greengrass in the *AWS IoT Greengrass Developer Guide*.

   We recommend that you run the AWS IoT Greengrass device setup script to quickly get started. If you want to review AWS IoT Greengrass requirements and processes more closely, you can walk through the steps in Module 1 and Module 2 to set up AWS IoT Greengrass.

   > **Important**
   >
   > Review the AWS Regions where AWS IoT SiteWise is supported. When you choose a Region for AWS IoT Greengrass, make sure that the Region also supports AWS IoT SiteWise. Otherwise, you can't connect your SiteWise Edge gateway to AWS IoT SiteWise.

   Before you continue to the next step, you should have AWS IoT Greengrass Core software installed on your SiteWise Edge gateway.

5. Run the following commands to install Java 8.

   ```
   sudo apt update
         sudo apt install openjdk-8-jre
   ```

   The SiteWise Edge gateway software that you install later in this guide uses a Java 8 runtime.

6. Run the following command to verify that Java installed successfully.

   ```
   java -version
   ```

7. The AWS IoT Greengrass Core software assumes a `java8` directory. Run the following command to link your Java installation to that `java8` directory.

```
sudo ln -s /usr/bin/java /usr/bin/java8
```

8. Run the following command to create a `/var/sitewise` data directory and give the `ggc_user` permissions for that directory. AWS IoT SiteWise stores data in this directory. You created the `ggc_user` when you set up AWS IoT Greengrass earlier in this procedure.

```
sudo mkdir /var/sitewise
      sudo chown ggc_user /var/sitewise
      sudo chmod 700 /var/sitewise
```

   The `/var/sitewise` is the default directory that AWS IoT SiteWise uses. You can customize the directory path (for example, replace `/var/sitewise` with `/var/`*`custom`*`/`*`path`*`/`), but doing so requires extra steps after the SiteWise Edge gateway is created. For more information, see step 6 in Configuring the AWS IoT SiteWise connector.

9. If needed, ask your IT administrator to add the following endpoints and ports to your local network allow list:

   - Ports: 443, 8443, and 8883

     > **Important**
     >
     > You can configure AWS IoT Greengrass Core to use only port 443 for all network communications. For more information, see Connect on port 443 or through a network proxy in the *AWS IoT Greengrass Developer Guide*.

   - The IP address of your SiteWise Edge gateway (port 443). To obtain the IP address, run the `ip address` or `ifconfig` command and note the `inet` value (for example, `203.0.113.0`).
   - The AWS IoT SiteWise data endpoint: `data.iotsitewise.`*`region`*`.amazonaws.com` (port 443).
   - The following AWS endpoints that the SiteWise Edge gateway uses. You can find these in the `/`*`greengrass-root`*`/config/config.json` file. Replace *`greengrass-root`* with the root of your AWS IoT Greengrass installation.
     - *ggHost*: `greengrass-ats.iot.`*`region`*`.amazonaws.com` (ports 443, 8443, and 8883).

- *iotHost*: *prefix*-ats.iot.*region*.amazonaws.com (ports 443, 8443, and 8883).

For more information, see AWS IoT Greengrass endpoints and quotas.

10. If the AWS IoT Greengrass Core software isn't already running, run the following command to start the AWS IoT Greengrass Core software. Replace *greengrass-root* with the root of your AWS IoT Greengrass installation. The default *greengrass-root* is /greengrass.

```
cd /greengrass-root/ggc/core
      sudo ./greengrassd start
```

You should see this message: Greengrass successfully started with PID: *some-PID-number*

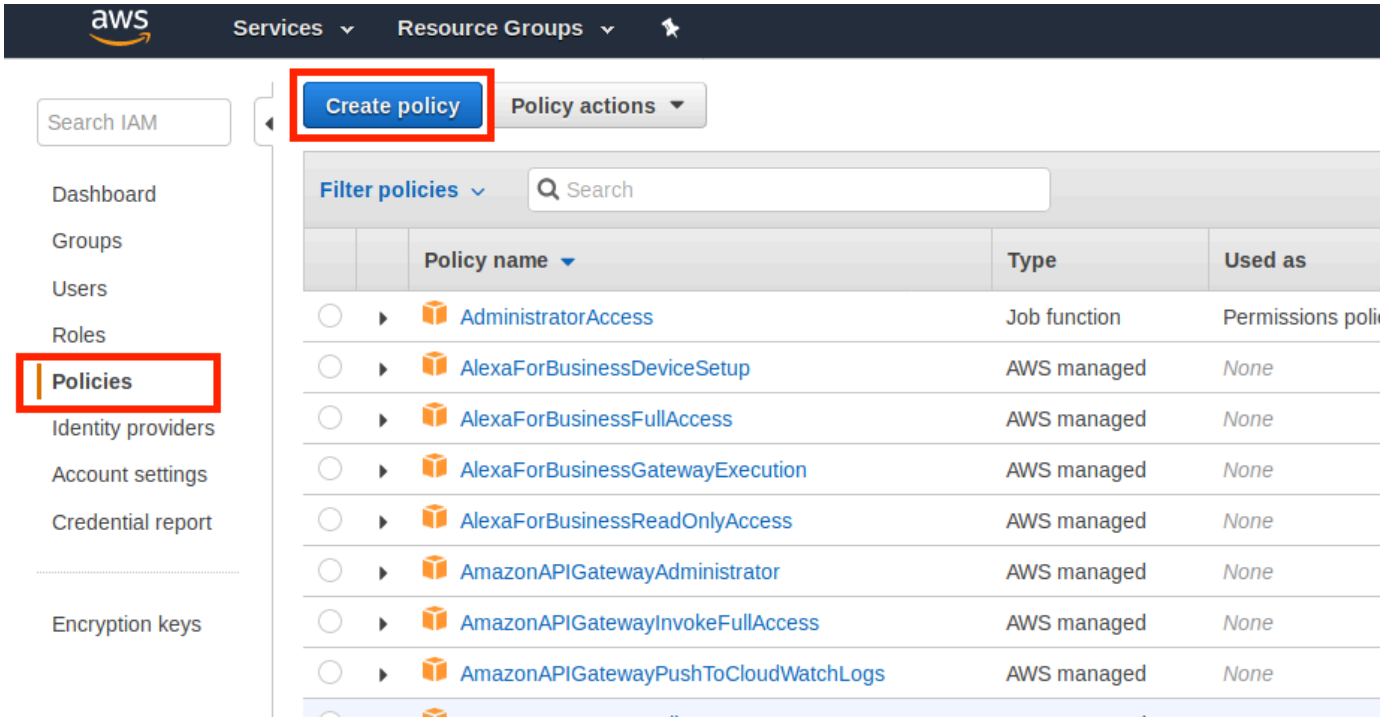
11. Configure the AWS IoT Greengrass Core software to automatically start when your SiteWise Edge gateway turns on. Consult the documentation for your SiteWise Edge gateway's operating system.

Creating an IAM policy and role

You must create an AWS Identity and Access Management (IAM) policy and role to allow the SiteWise Edge gateway to access AWS IoT SiteWise on your behalf.

To create an IAM policy and role

1. Navigate to the IAM console.
2. In the navigation pane, choose **Policies**, and then choose **Create policy**.



3. On the **JSON** tab, delete the current contents of the policy field, and paste the following policy into the field.

```
{
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Action": "iotsitewise:BatchPutAssetPropertyValue",
            "Resource": "*"
          }
        ]
      }
```

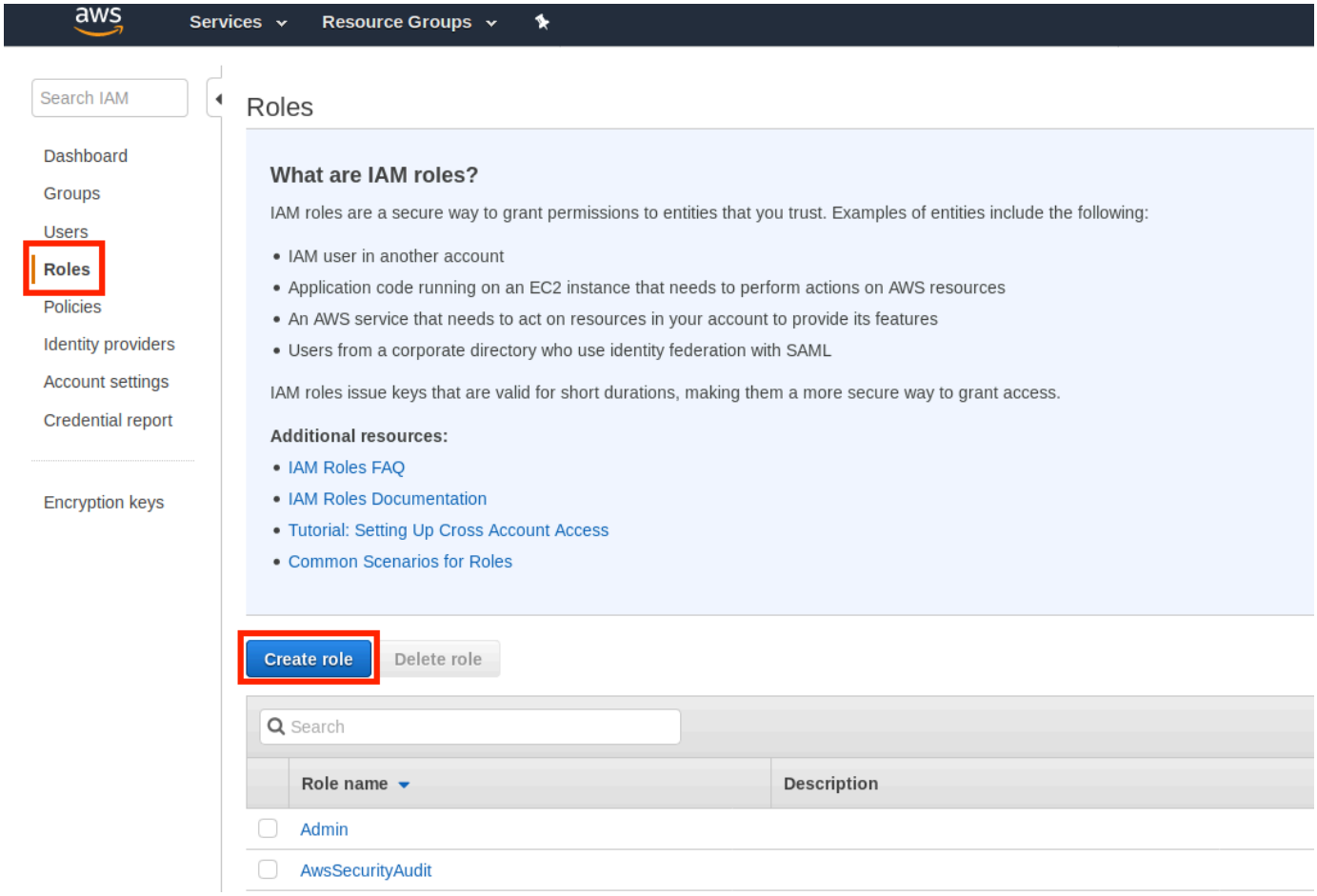
Note

To improve security, you can specify an AWS IoT SiteWise asset hierarchy path in the `Condition` property. The following example is a trust policy that specifies an asset hierarchy path.

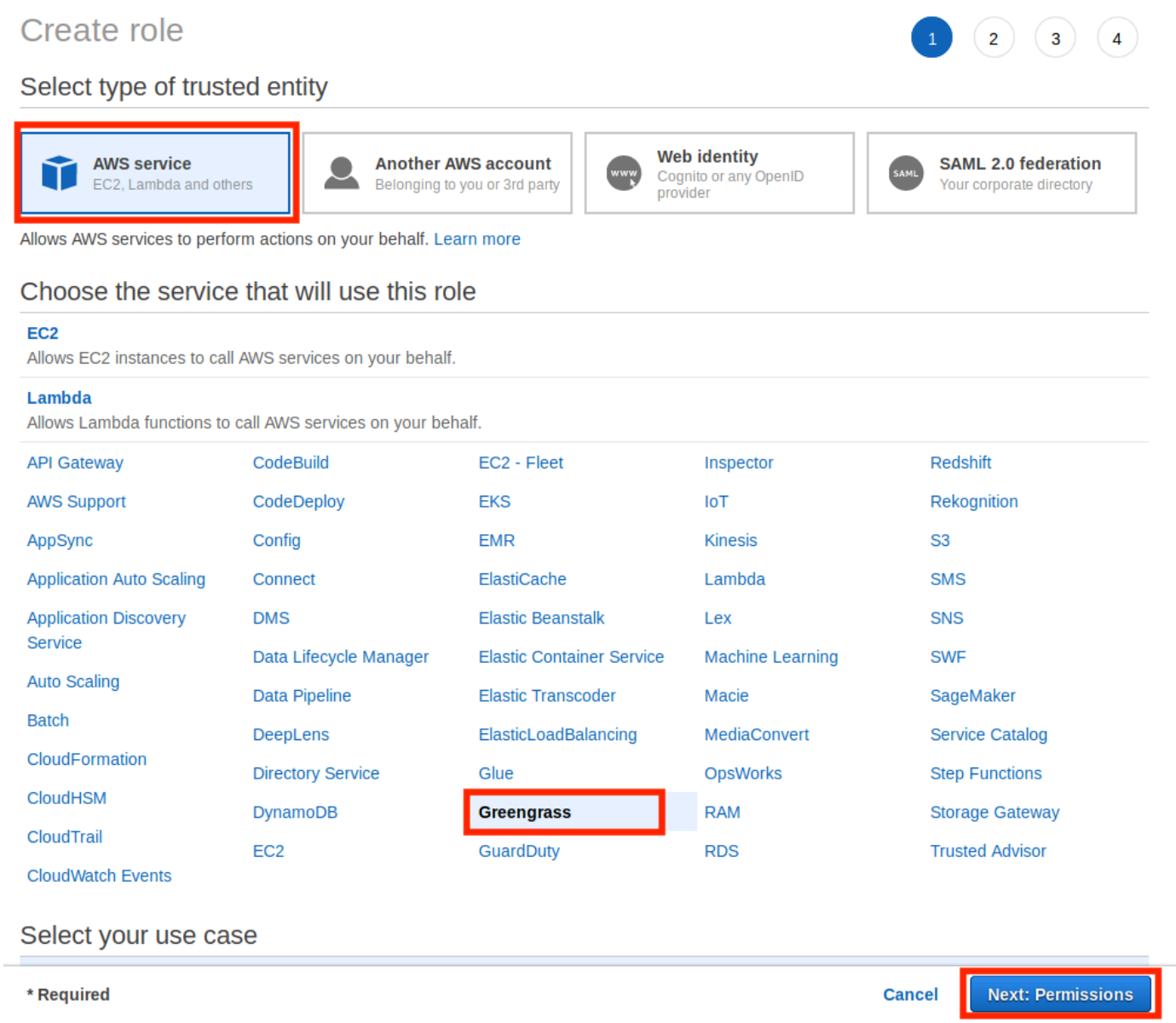
```
{
        "Version": "2012-10-17",
        "Statement": [
```

```
        {
          "Effect": "Allow",
          "Action": "iotsitewise:BatchPutAssetPropertyValue",
          "Resource": "*",
          "Condition": {
            "StringLike": {
              "iotsitewise:assetHierarchyPath": [
                "/root node asset ID",
                "/root node asset ID/*"
              ]
            }
          }
        }
      ]
    }
```

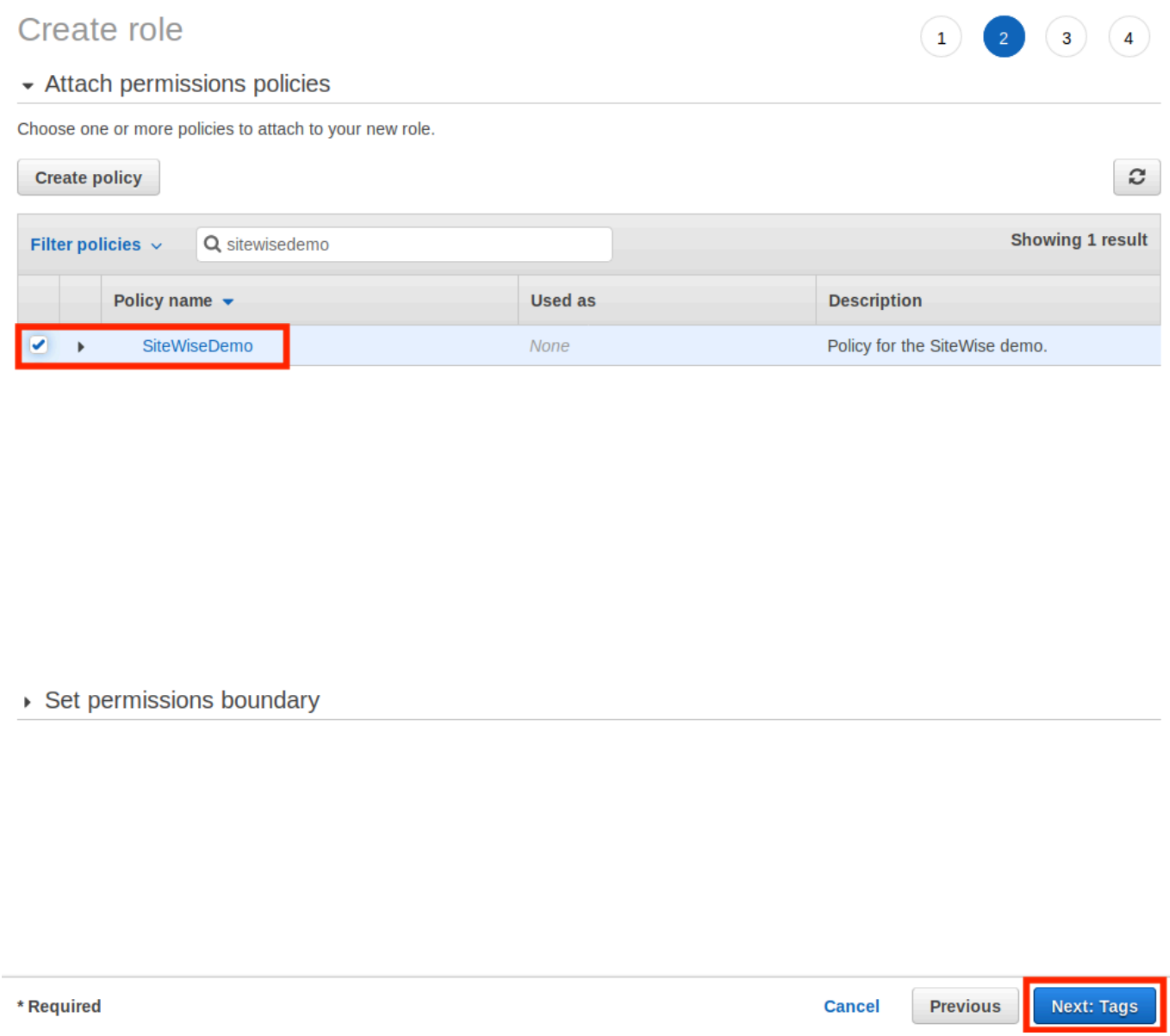
4. Choose **Review policy**.
5. Enter a name and description for the policy, and then choose **Create policy**.
6. In the navigation pane, choose **Roles**, and then choose **Create role**.



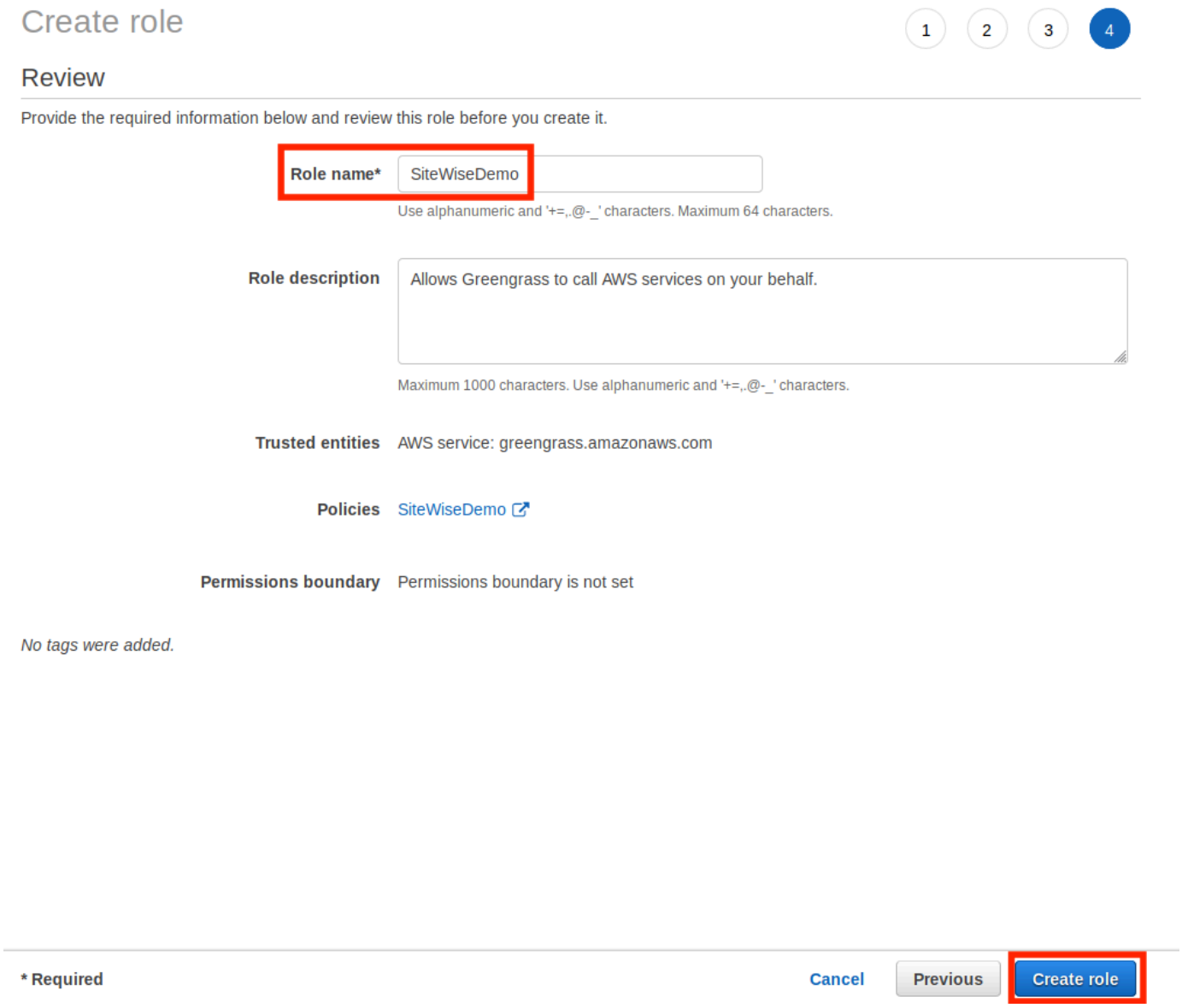
7. Under **Select type of trusted entity**, choose **AWS service**. Under **Choose the service that will use the role**, choose **Greengrass** as the service that will use the role, and then choose **Next: Permissions**.



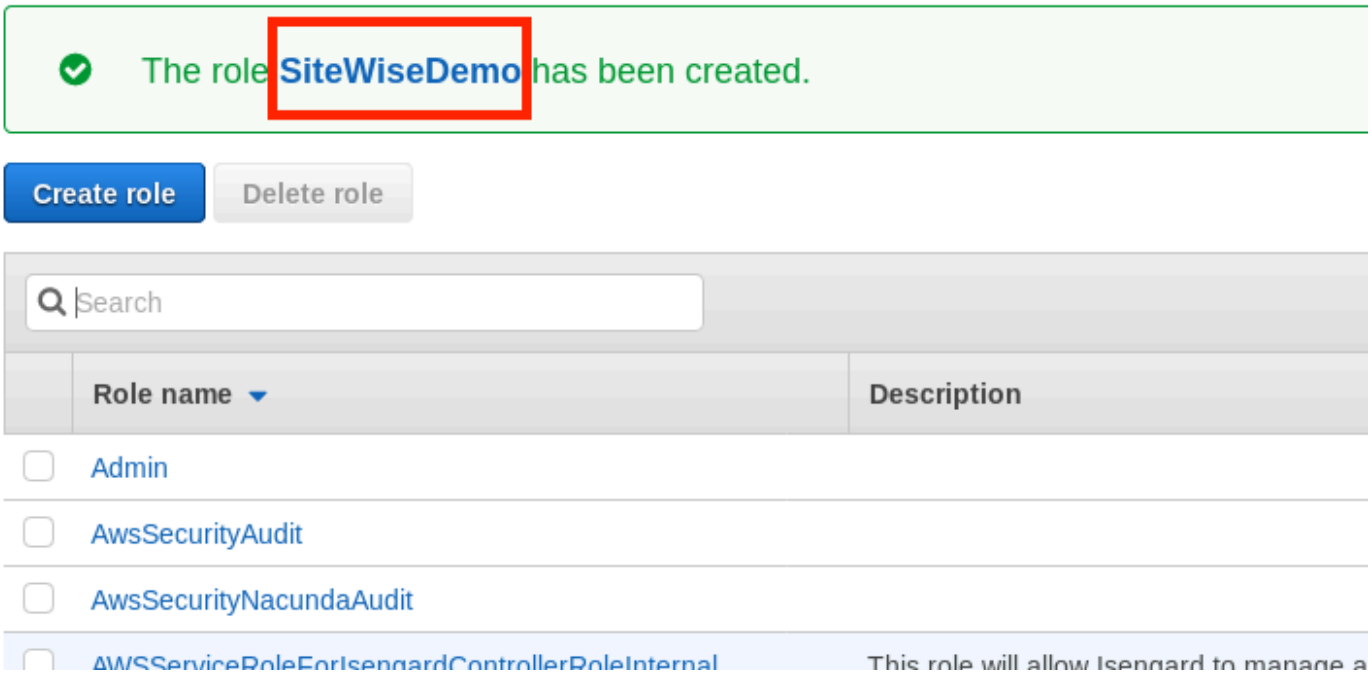
8. Search for the policy that you created, select the check box, and then choose **Next: Tags**.



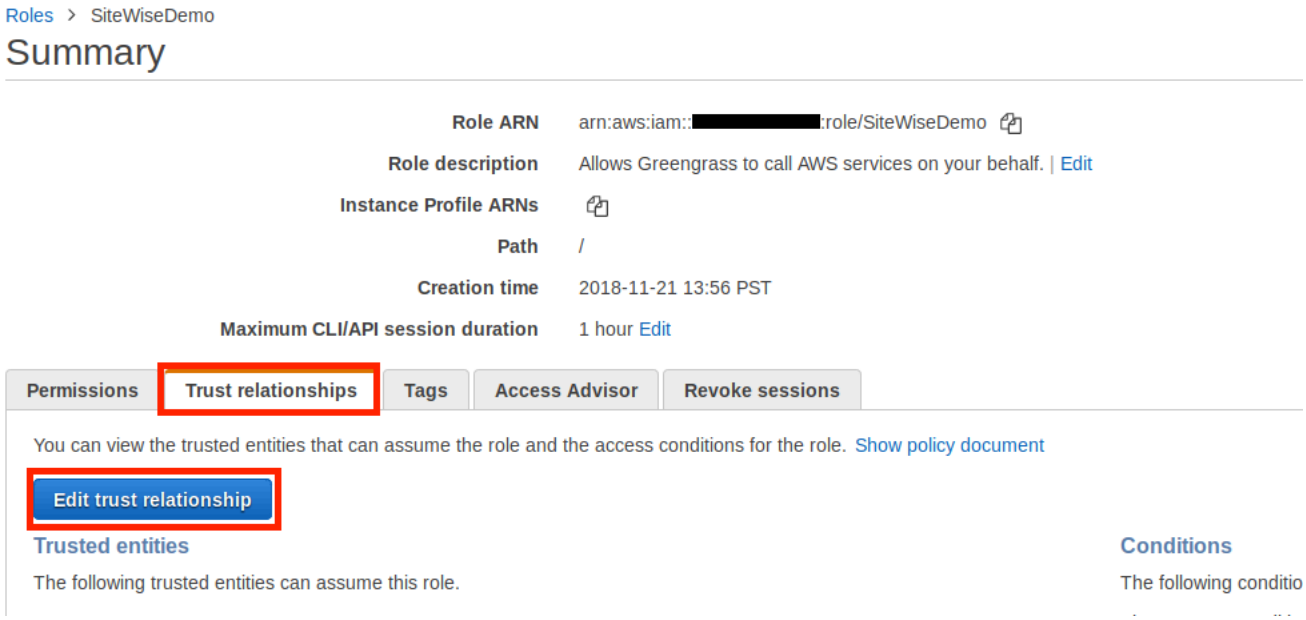
9. (Optional) Add tags to your role, and then choose **Next: Review**.
10. Enter a name and description for the role, and then choose **Create role**.



11. In the green banner, choose the link to your new role. You can also use the search field to find the role.



12. Choose the **Trust relationships** tab, and then choose **Edit trust relationship**.



13. Replace the current contents of the policy field with the following, and then choose **Update Trust Policy**.

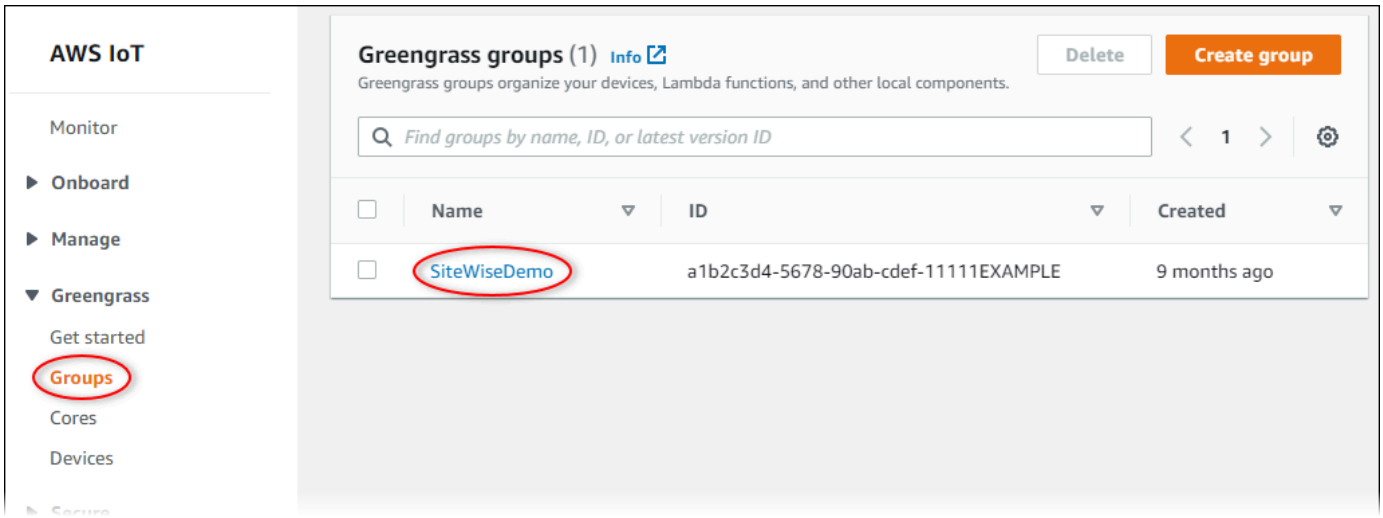
```
{
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
```

```
            "Principal": {
              "Service": "greengrass.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      }
```

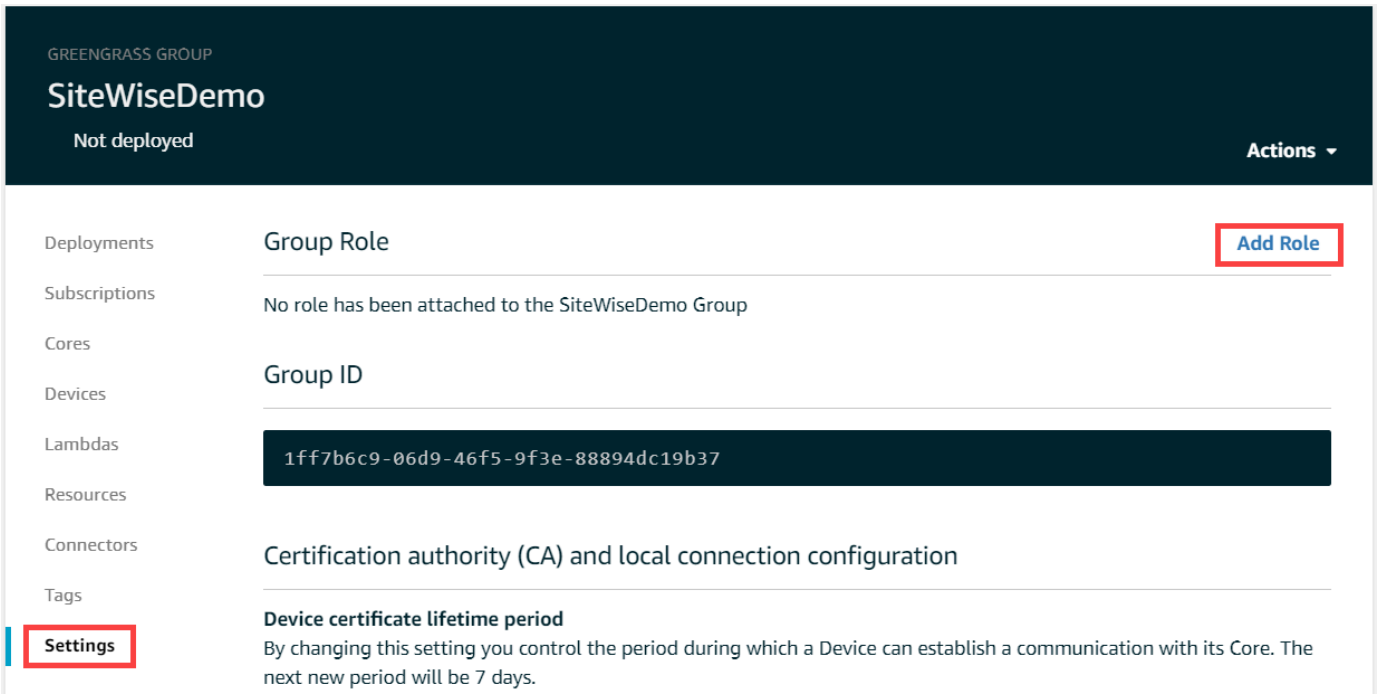
Configuring an AWS IoT Greengrass group

To attach an IAM role to a group and enable stream manager

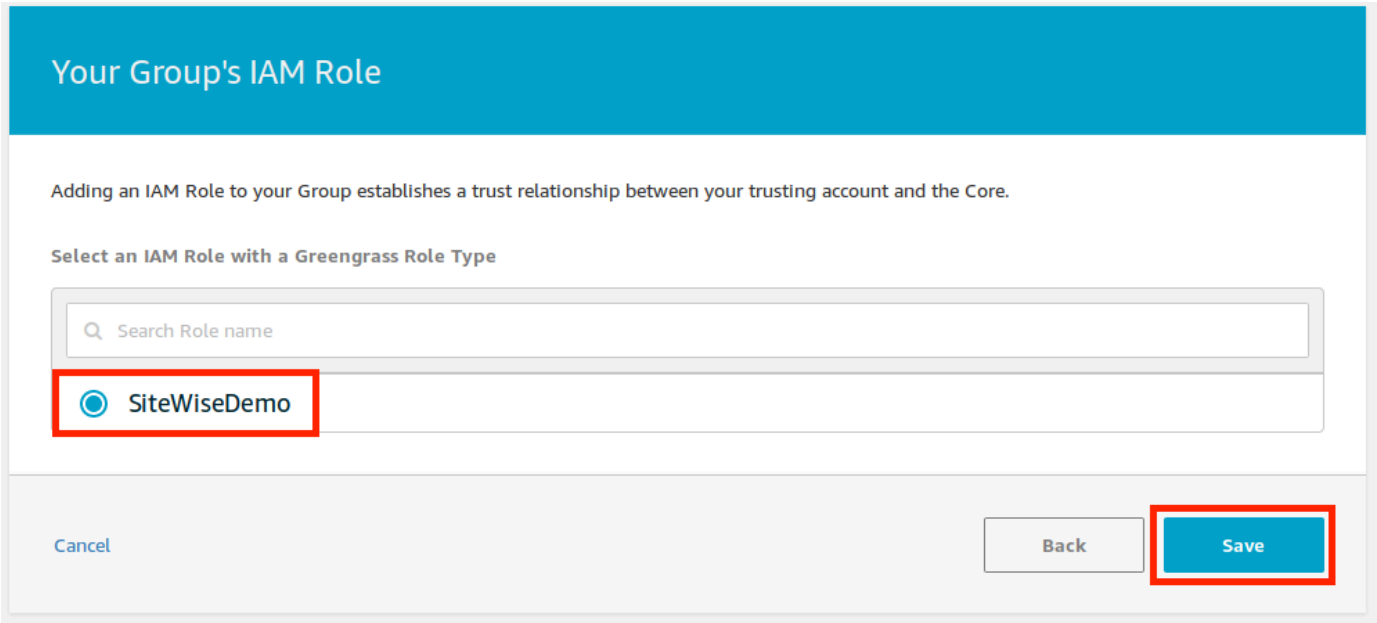
1. Navigate to the AWS IoT Greengrass console.
2. In the left navigation pane, under **Greengrass**, choose **Groups**, and then choose the group that you created in Setting up the SiteWise Edge gateway environment.



3. In the left navigation pane, choose **Settings**. In the **Group Role** section, choose **Add Role**.

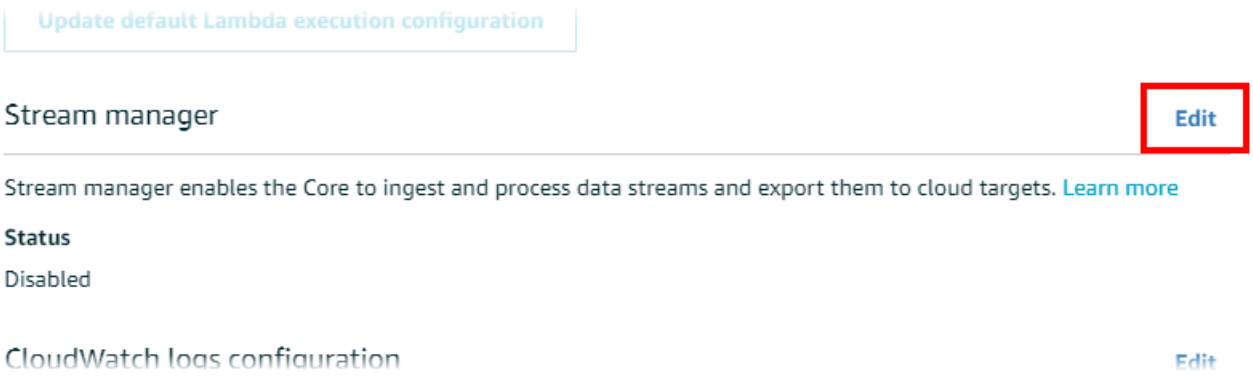


4. Choose the role that you created in Creating an IAM policy and role, and then choose **Save**.



5. On the **Settings** page, in the **Stream manager** section, choose **Edit**.

   Stream manager is a feature of AWS IoT Greengrass that enables your AWS IoT Greengrass Core to stream data to the AWS Cloud. SiteWise Edge gateways require that stream manager is enabled. For more information, see Manage data streams on the AWS IoT Greengrass Core in the *AWS IoT Greengrass Version 1 Developer Guide*.



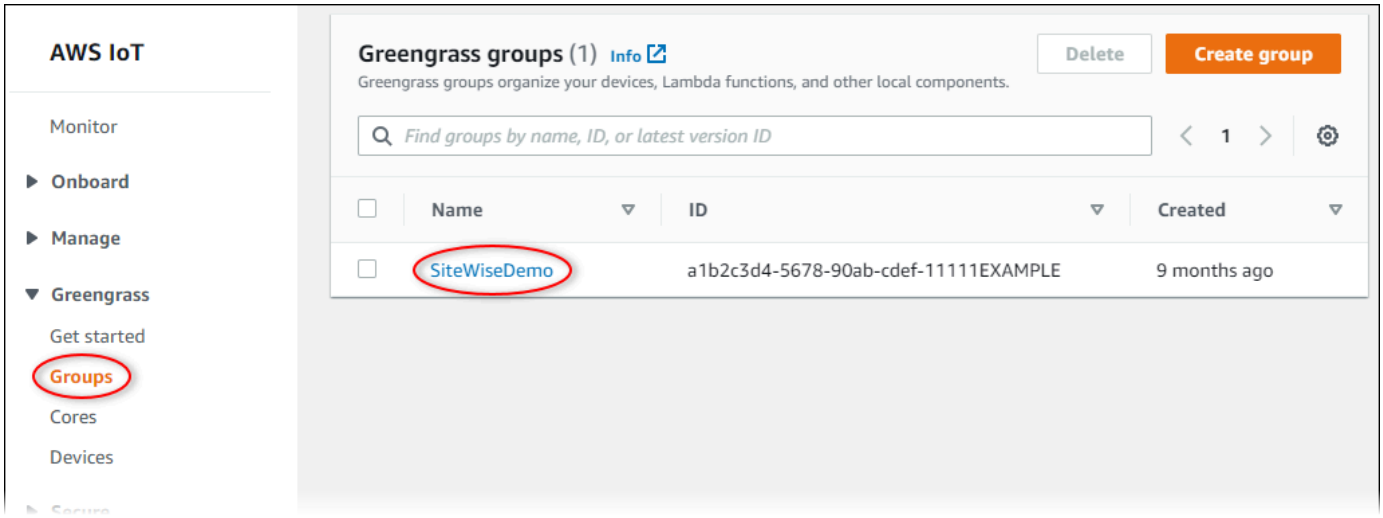
6. Choose **Enable**, and then choose **Save**.
7. In the upper-left corner, choose **Services** to prepare for the next procedure.

Configuring the AWS IoT SiteWise connector

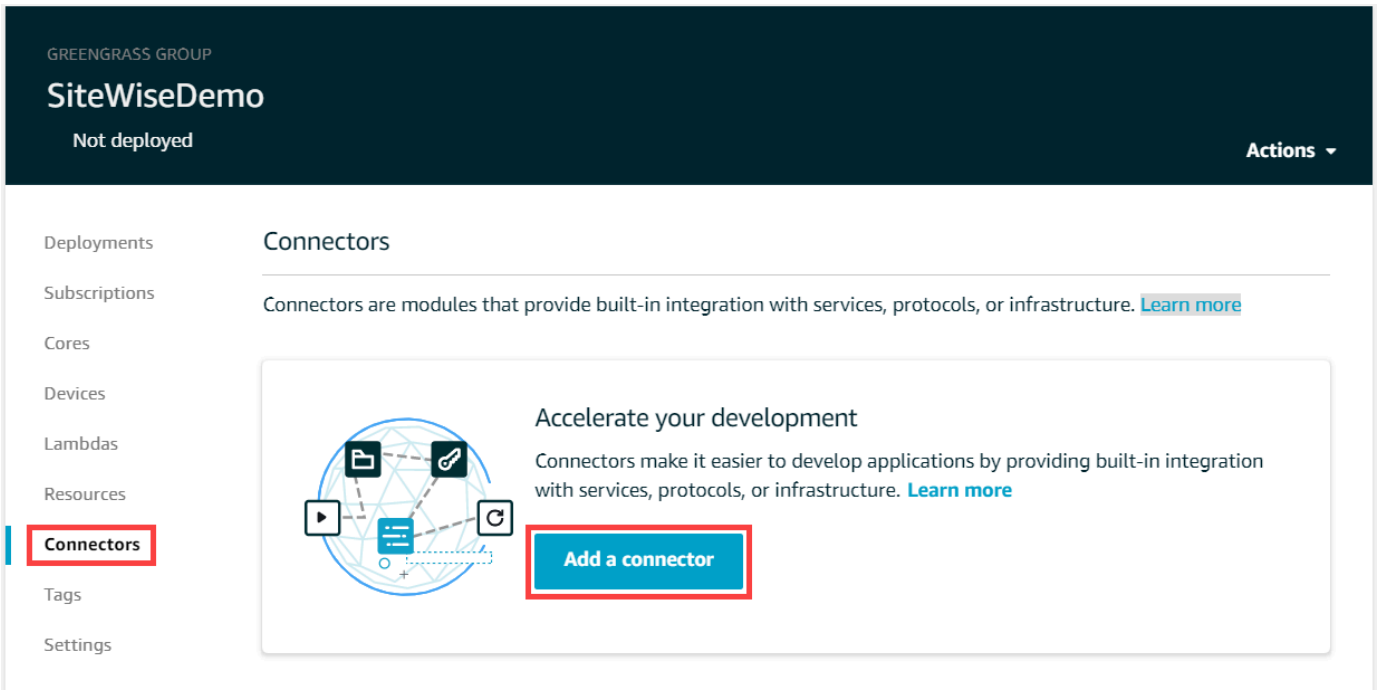
In this procedure, you configure the AWS IoT SiteWise connector on your Greengrass group. Components are prebuilt modules that accelerate the development lifecycle for common edge scenarios. For more information, see AWS IoT Greengrass connectors in the *AWS IoT Greengrass Version 1 Developer Guide*.

To configure the AWS IoT SiteWise connector

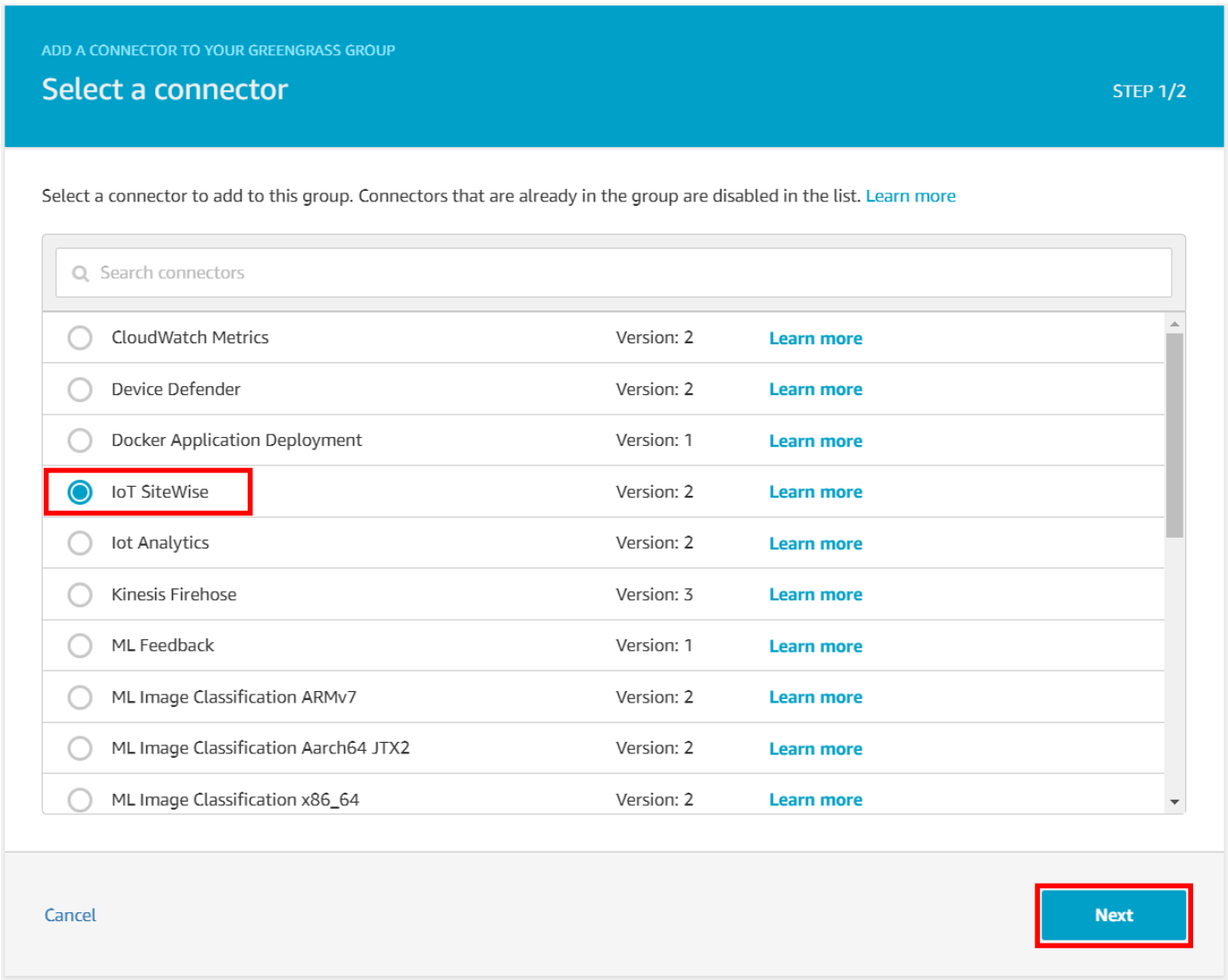
1. Navigate to the AWS IoT Greengrass console.
2. In the left navigation pane, under **Greengrass**, choose **Groups**, and then choose the group that you created in Setting up the SiteWise Edge gateway environment.



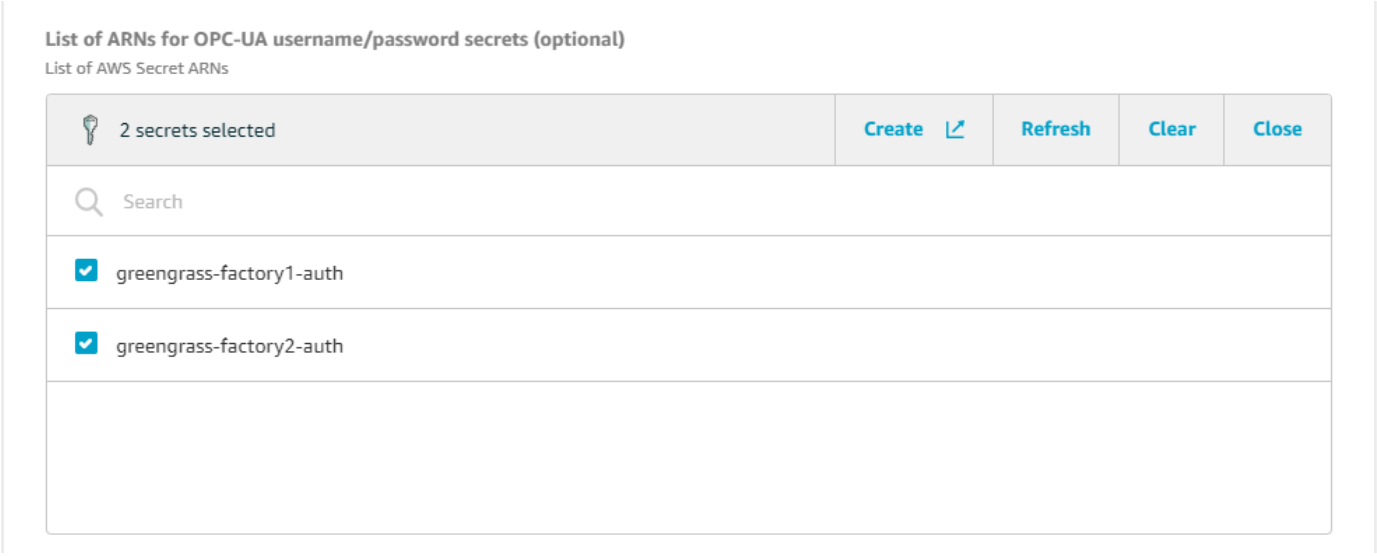
3. In the left navigation page, choose **Connectors**. On the **Connectors** page, choose **Add a connector**.



4. Choose **IoT SiteWise** from the list and choose **Next**.



5. If your server requires authentication, you can create AWS Secrets Manager secrets with the server's user name and password. Then, you can attach each secret to your Greengrass group and choose them under **List of ARNs for username/password secrets**. For more information about how to create and configure secrets, see Configuring source authentication. You can also add secrets to your connector later.



6. If you set up your SiteWise Edge gateway with a different path than `/var/sitewise`, enter that path for **Local storage path**.
7. (Optional) Enter a maximum disk buffer size for the connector. If the AWS IoT Greengrass core loses connection to the AWS Cloud, the connector caches data until it can successfully connect. If the cache size exceeds the maximum disk buffer size, the connector discards the oldest data from the queue.
8. Choose **Add**.
9. In the upper-right corner, in the **Actions** menu, choose **Deploy**.
10. Choose **Automatic detection** to start the deployment.

    If the deployment fails, choose **Deploy** again. If the deployment continues to fail, see AWS IoT Greengrass deployment troubleshooting.

Adding the SiteWise Edge gateway to AWS IoT SiteWise

In this procedure, you add your SiteWise Edge gateway's Greengrass group to AWS IoT SiteWise. After you register your SiteWise Edge gateway with AWS IoT SiteWise, the service can deploy your data source configurations to your SiteWise Edge gateway.

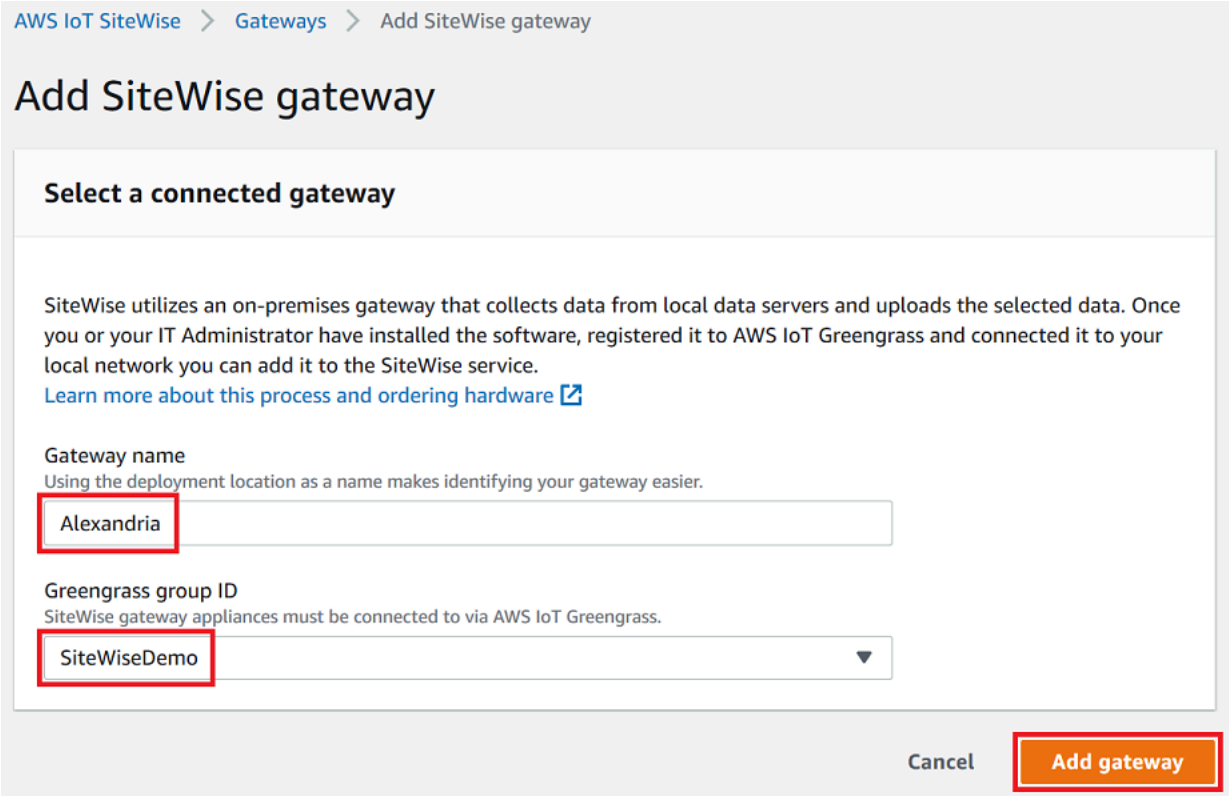
To add the SiteWise Edge gateway to AWS IoT SiteWise

1. Navigate to the AWS IoT SiteWise console.
2. Choose **Add gateway**.
3. On the **Add SiteWise gateway** page, do the following:

a. Enter a **Name** for the SiteWise Edge gateway. Consider including the location of the SiteWise Edge gateway in the name so that you can easily identify it.

b. For **Greengrass group ID**, choose the Greengrass group that you created earlier.

Example



c. (Optional) For **Edge capabilities**, choose **Data processing pack**. This enables communication between your SiteWise Edge gateway and any asset models and assets configured for the edge. For more information, see the section called “Enabling edge data processing”.

> **Important**
>
> If you add the data processing pack to your SiteWise Edge gateway, you must configure and deploy the SiteWise Edge connector on your AWS IoT Greengrass group. Follow the next steps.

d. Choose **Add gateway**.

4. If you add the data processing pack to your SiteWise Edge gateway, configure and deploy the AWS IoT SiteWise Data Processor connector on your AWS IoT Greengrass group. Follow the

steps in the section called "Configuring the AWS IoT SiteWise connector" to configure the AWS IoT SiteWise Data Processor connector:

a. For **Select a connector** in the AWS IoT Greengrass console, choose **AWS IoT SiteWise Data Processor**.

b. For **Local storage path**, enter the path to your SiteWise Edge gateway.

c. Choose **Add**.

d. In the upper-right corner, in the **Actions** menu, choose **Deploy**, and then choose **Automatic detection** to start the deployment.

After your SiteWise Edge gateway deploys, you can add a source for each piece of industrial equipment from which you want your SiteWise Edge gateway to ingest data. For more information, see Configuring data sources.

You can view Amazon CloudWatch metrics to verify that your SiteWise Edge gateway connects to AWS IoT SiteWise. For more information, see AWS IoT Greengrass Version 1 gateway metrics.

Configuring data sources on AWS IoT Greengrass V1 SiteWise Edge gateways

After you set up an AWS IoT SiteWise Edge gateway, you can configure data sources so that your SiteWise Edge gateway can ingest data from local industrial equipment to AWS IoT SiteWise. Each source represents a local server, such as an OPC-UA server, that your SiteWise Edge gateway connects and retrieves industrial data streams. For more information about setting up a SiteWise Edge gateway, see Configuring a AWS IoT Greengrass V1 SiteWise Edge gateway.

> **Note**
>
> AWS IoT SiteWise restarts your SiteWise Edge gateway each time you add or edit a source. Your SiteWise Edge gateway won't ingest data while it's restarting. The time to restart your SiteWise Edge gateway depends on the number of tags on your SiteWise Edge gateway's sources. Restart time can range from a few seconds (for a SiteWise Edge gateway with few tags) to several minutes (for a SiteWise Edge gateway with many tags).

After you create sources, you can associate your data streams with asset properties. For more information about how to create and use assets, see Modeling industrial assets and Mapping industrial data streams to asset properties.

You can view CloudWatch metrics to verify that a data source is connected to AWS IoT SiteWise. For more information, see AWS IoT Greengrass Version 1 gateway metrics.

Currently, AWS IoT SiteWise supports the following data source protocols:

- OPC-UA – A machine-to-machine (M2M) communication protocol for industrial automation.
- Modbus TCP – A data communications protocol used to interface with programmable logic controllers (PLCs).
- Ethernet/IP (EIP) – An industrial network protocol that adapts the Common Industrial Protocol (CIP) to standard Ethernet.

> **Note**
>
> SiteWise Edge gateways running on AWS IoT Greengrass V2 currently don't support Modbus TCP and Ethernet IP sources.

Topics

Configure a Modbus TCP source

You can use the AWS IoT SiteWise console or an AWS IoT SiteWise Edge gateway capability to define and add a Modbus TCP source to your SiteWise Edge gateway. This source represents a local Modbus TCP server.

Note

- SiteWise Edge gateways running on AWS IoT Greengrass V2 currently don't support Modbus TCP sources.
- You must install the AWS IoT SiteWise connector to use a Modbus TCP source.

You can use the Modbus TCP source to convert the data type from your source into a different data type when it's received on your SiteWise Edge gateway. The source data type determines the data types that you can choose for your destination data. You can also choose to swap bytes using the Modbus TCP source. The following table provides more information on the source data types, destination data types, and swap modes that are compatible.

For more information about swap modes, see the How Real (Floating Point) and 32-bit Data is Encoded in Modbus RTU Messages article on Modbus message encoding.

Source data type	Compatible destination data types	Compatible swap modes	Compatible connector versions
ASCII	String	noSwap	2
UTF8	String	noSwap	2
ISO8859	String	noSwap	2
Int16	Integer, Double, String	noSwap	1 and 2
Int32	Integer, Double, String	noSwap, byteWordSwap, byteSwap, wordSwap	1 and 2
Float	Double, String	noSwap, byteWordSwap, byteSwap, wordSwap	1 and 2
Boolean	Boolean	noSwap	1 and 2

Source data type	Compatible destination data types	Compatible swap modes	Compatible connector versions
Hex-dump	String	noSwap	1 and 2

Topics

- Configure a Modbus TCP source (console)
- Configure a Modbus TCP source (CLI)

Configure a Modbus TCP source (console)

To configure a Modbus TCP source

1. Navigate to the AWS IoT SiteWise console.
2. In the left navigation pane, choose **Gateways**.
3. On the SiteWise Edge gateway you want to create a source for, choose **Manage**, and then choose **View details**.
4. Choose **New source** in the upper-right corner.
5. For **Protocol options**, choose **Modbus TCP**.
6. For **Modbus TCP source configuration**, enter a **Name** for the source.
7. For **IP address**, enter the IP address for the data source server.
8. (Optional) Enter the **Port** and **Unit ID** for the source server.
9. (Optional) For **Minimum inter-request duration**, enter the time interval between subsequent requests sent to your server. Your SiteWise Edge gateway automatically calculates the minimum allowable interval based on your device and the number of registers you have.
10. For **Property groups**, enter a **Name**.
11. For **Properties**:
    a. For **Tag**, enter a property alias for your register set. For example, **`TT-001`**.
    b. For **Register address**, enter the register address that starts the register set.
    c. For **Source data type**, choose the Modbus TCP data type you want to convert data from. This defaults to **Hex dump**.

> **Note**
>
> The source data type you choose determines the data size, destination data type, and swap mode you can choose. For more information, see the section called "Configure a Modbus TCP source".

d. For **Data size**, enter the number of registers to read when starting from the **Register address**. This is determined by the source data type you choose for this source.

e. For **Destination data type**, choose the AWS IoT SiteWise data type that you want your data to be converted to. The default is **String**. The destination type must be compatible with the source data type you choose for this source. For more information, see the section called "Configure a Modbus TCP source".

f. For **Swap mode**, choose the data swap mode you want to use to read data from your register set. The swap mode must be compatible with the source data type you choose for this source. For more information, see the section called "Configure a Modbus TCP source".

12. For **Scan rate**, update the rate at which you want the SiteWise Edge gateway to read your registers. AWS IoT SiteWise automatically calculates the minimum allowable scan rate for your SiteWise Edge gateway.

13. (Optional) For **Destination**, choose where the source data is sent. By default, your source sends data to AWS IoT SiteWise.You can use a AWS IoT Greengrass stream to export your data to a local destination or to the AWS Cloud instead.

> **Note**
>
> You must choose AWS IoT SiteWise as the destination for your source data if you want to process data from this source at the edge with AWS IoT SiteWise. For more information about processing data at the edge, see the section called "Enabling edge data processing".

To send your data to another destination:

a. For **Destination options**, choose **Other destinations**.

b. For **Greengrass stream name**, enter the exact name of your AWS IoT Greengrass stream.

> **Note**
>
> You can use a stream that you've already created, or you can create a new AWS IoT Greengrass stream to export your data. If you want to use an existing stream, you must enter the exact name of the stream or a new stream will be created.
> For more information about working with AWS IoT Greengrass streams, see Manage data streams in the AWS IoT Greengrass developer guide.

14. Choose **Add source**.

    AWS IoT SiteWise deploys the SiteWise Edge gateway configuration to your AWS IoT Greengrass core. You don't need to manually launch a deployment.

Configure a Modbus TCP source (CLI)

You can define Modbus TCP data sources in a SiteWise Edge gateway capability. You must define all of your Modbus TCP sources in a single capability configuration.

For more information about defining sources with the AWS CLI, see the section called “Configuring data sources (AWS CLI)”.

> **Note**
>
> You must install the AWS IoT SiteWise connector to use a Modbus TCP source.

This capability has the following versions.

Version	Namespace
1	`iotsitewise:modbuscollector:1`

Modbus TCP capability configuration parameters

When you define Modbus TCP sources in a capability configuration, you must specify the following information in the `capabilityConfiguration` JSON document:

sources

A list of Modbus-TCP source definition structures that each contain the following information:

name

A unique, friendly name for the source.

measurementDataStreamPrefix

(Optional) A string to prepend to all data streams from the source. The SiteWise Edge gateway adds this prefix to all data streams from this source. Use a data stream prefix to distinguish between data streams that have the same name from different sources. Each data stream should have a unique name within your account.

destination

A destination structure that contains the following information:

type

The type of the destination.

streamName

The name of the AWS IoT Greengrass stream.

streamBufferSize

The size of the stream buffer.

endpoint

An endpoint structure that contains the following information:

ipAddress

The IP address of the Modbus TCP source.

port

(Optional) The port of the Modbus TCP source.

unitId

(Optional) The unitId. This defaults to a value of 1.

minimumInterRequestDuration

The minimum duration between each request in milliseconds.

propertyGroups

The list of property groups that define the tag definition requested by the protocol.

name

The name of the property group. This should be a unique identifier.

tagPathDefinitions

The location of the measurement within the source. For example, the byte and word order, address, and transformation type. The structure of each `MeasurementPathDefinition` is defined by the connector.

scanMode

Defines the scan mode behavior and configurable parameters for the source.

Configure an Ethernet/IP (EIP) source

You can use the AWS IoT SiteWise console or a SiteWise Edge gateway capability to define and add an Ethernet IP source to your SiteWise Edge gateway. This source represents a local Ethernet IP server.

> **Note**
>
> - SiteWise Edge gateways running on AWS IoT Greengrass V2 currently don't support Ethernet IP sources.
> - You must install the AWS IoT SiteWise connector to use an Ethernet IP source.

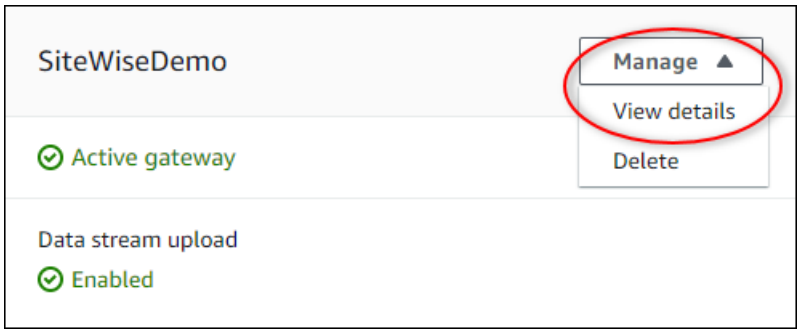
Topics

Configure an Ethernet/IP source (console)

To configure an Ethernet/IP source

1. Navigate to the AWS IoT SiteWise console.

2. In the left navigation pane, choose **Gateways**.
3. On the SiteWise Edge gateway you want to create a source for, choose **Manage**, and then choose **View details**.



4. Choose **New source** in the upper-right corner.
5. For **Protocol options**, choose **Ethernet/IP (EIP)**.
6. For **EtherNet/IP source configuration**, enter a **Name** for the source.
7. For **IP address**, enter the IP address for the data source server.
8. (Optional) Enter the **Port** for the source server.
9. For **Minimum inter-request duration**, enter the time interval between subsequent requests sent to your server. Your SiteWise Edge gateway automatically calculates the minimum allowable interval based on your device and the number of registers you have.
10. For **Property groups**, enter a **Name**.
11. For **Properties**:
    a. For **Tag**, enter the property alias for your register set. For example, **`boiler.inlet.temperature.value`**.
    b. For **Destination data type**, choose the AWS IoT SiteWise data type that you want your data to be converted to. The default is **String**.
12. For **Scan rate**, update the rate at which you want the SiteWise Edge gateway to read your registers. AWS IoT SiteWise automatically calculates the minimum allowable scan rate for your SiteWise Edge gateway.
13. (Optional) For **Destination**, choose where the source data is sent. By default, your source sends data to AWS IoT SiteWise.You can use a AWS IoT Greengrass stream to export your data to a local destination or to the AWS Cloud instead.

> **Note**
>
> You must choose AWS IoT SiteWise as the destination for your source data if you want to process data from this source at the edge with AWS IoT SiteWise. For more information about processing data at the edge, see the section called "Enabling edge data processing".

To send your data to another destination:

a. For **Destination options**, choose **Other destinations**.

b. For **Greengrass stream name**, enter the exact name of your AWS IoT Greengrass stream.

> **Note**
>
> You can use a stream that you've already created, or you can create a new AWS IoT Greengrass stream to export your data. If you want to use an existing stream, you must enter the exact name of the stream or a new stream will be created.
> For more information about working with AWS IoT Greengrass streams, see Manage data streams in the AWS IoT Greengrass developer guide.

14. Choose **Add source**.

AWS IoT SiteWise deploys the SiteWise Edge gateway configuration to your AWS IoT Greengrass core. You don't need to manually launch a deployment.

Configure an Ethernet/IP source (CLI)

You can define EIP data sources in a SiteWise Edge gateway capability. You must define all of your EIP sources in a single capability configuration.

For more information about defining sources with the AWS CLI, see the section called "Configuring data sources (AWS CLI)".

> **Note**
>
> You must install the AWS IoT SiteWise connector to use an Ethernet IP source.

This capability has the following versions.

Version	Namespace
1	`iotsitewise:eipcollector:1`

EIP capability configuration parameters

When you define EIP sources in a capability configuration, you must specify the following information in the `capabilityConfiguration` JSON document:

sources

A list of EIP source definition structures that each contain the following information:

name

A unique, friendly name for the source. This can be up to 256 characters.

destinationPathPrefix

(Optional) A string to prepend to all data streams from the source. The SiteWise Edge gateway adds this prefix to all data streams from this source. Use a data stream prefix to distinguish between data streams that have the same name from different sources. Each data stream should have a unique name within your account.

destination

A destination structure that contains the following information:

type

The type of the destination.

streamName

The name of the AWS IoT Greengrass stream.

streamBufferSize

The size of the stream buffer.

endpoint

An endpoint structure that contains the following information:

ipAddress

The IP address of the EIP source.

port

(Optional) The port of the EIP source. Accepted values are numbers between 1 and 65535.

minimumInterRequestDuration

(Optional) The minimum duration between each request in milliseconds.

propertyGroups

The list of property groups that define the tag definition requested by the protocol. Each source can have one property group.

name

The name of the property group. This should be a unique identifier with a maximum length of 256 characters.

tagPathDefinitions

The list of structures specifying the data to collect from the Ethernet/IP device and how to transform it for output.

type

The type of the `tagPathDefinition`. For example, `EIPTagPath`.

path

The path of the `tagPathDefinition`. Each tag in a path can be a maximum length of 40 characters and can start with a letter or an underscore. Tags can't contain consecutive or trailing underscores. The path is prefixed with any value of destinationPathPrefix.

dstDataType

The data type to output the tag data. Accepted values are `integer`, `double`, `string`, and `boolean`.

scanMode

Defines the scan mode behavior and configurable parameters for the source.

type

The type of the scan mode behavior. Accepted values are `POLL`.

rate

The rate in milliseconds that the connector should read tags from the Ethernet/IP source.

Configuring source authentication

If your OPC-UA servers require authentication credentials to connect, you can define a user name and password in a secret for each source in AWS Secrets Manager. Then, you add the secret to your Greengrass group and IoT SiteWise connector to make the secret available to your SiteWise Edge gateway. For more information, see Deploy secrets to the AWS IoT Greengrass core in the *AWS IoT Greengrass Version 1 Developer Guide*.

After a secret is available to your SiteWise Edge gateway, you can choose it when you configure a source. Then, the SiteWise Edge gateway uses the authentication credentials from the secret when it connects to the source. For more information, see Configuring data sources.

Topics

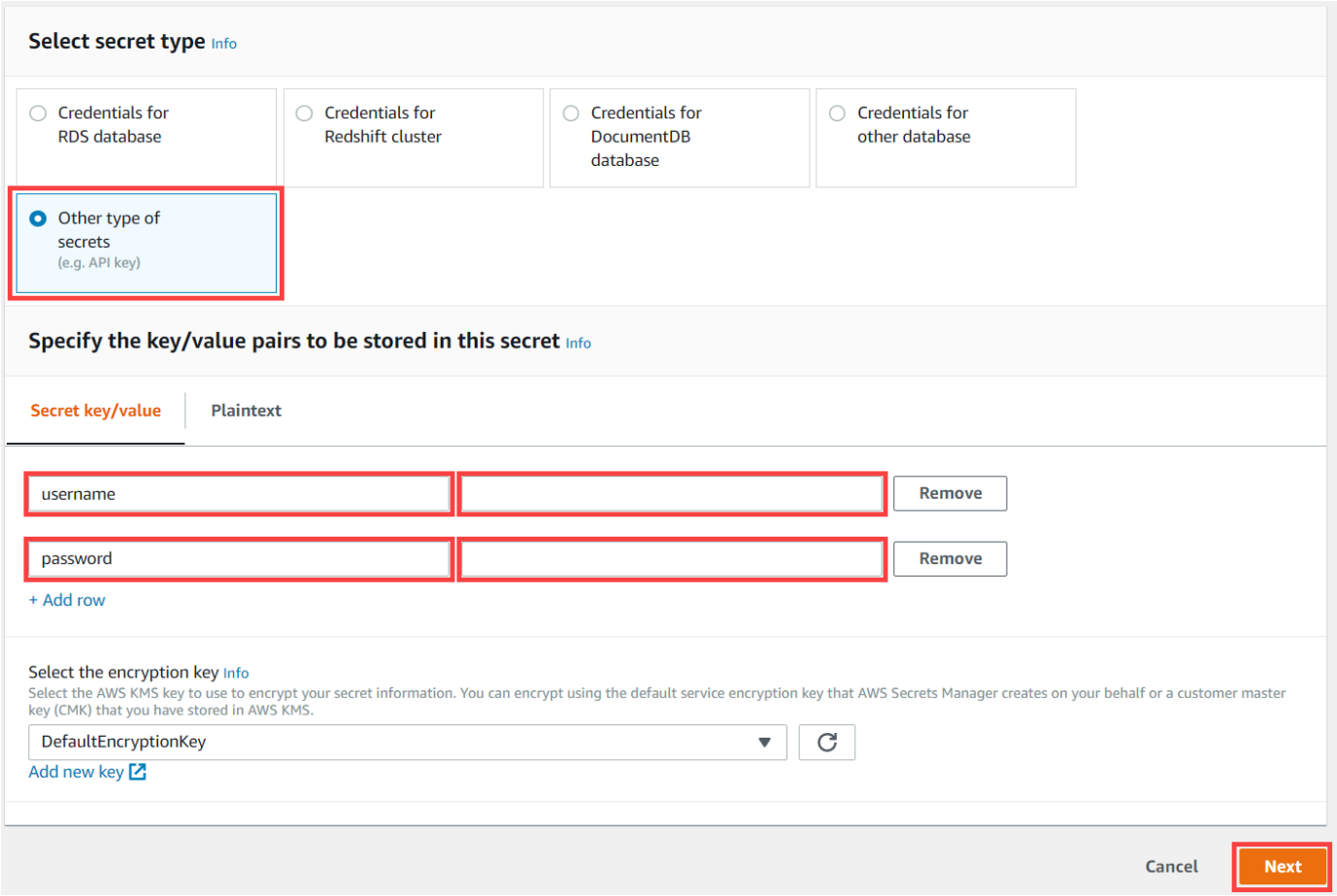
- Creating source authentication secrets
- Adding secrets to a Greengrass group
- Adding secrets to an IoT SiteWise connector

Creating source authentication secrets

In this procedure, you create an authentication secret for your source in Secrets Manager. In the secret, define **`username`** and **`password`** key-value pairs that contain authentication details for your source.

To create a source authentication secret

1. Navigate to the Secrets Manager console.
2. Choose **Store a new secret**.
3. Under **Select secret type**, choose **Other type of secrets**.
4. Enter **`username`** and **`password`** key-value pairs for your OPC-UA server's authentication values, and then choose **Next**.



5. Enter a **Secret name** that begins with `greengrass-`, such as **`greengrass-factory1-auth`**.

> **Important**
>
> You must use the `greengrass-` prefix for the default AWS IoT Greengrass service role to access your secrets. If you want to name your secrets without this prefix, you must grant AWS IoT Greengrass custom permissions to access your secrets. For more information, see Allow AWS IoT Greengrass to get secret values in the *AWS IoT Greengrass Version 1 Developer Guide*.

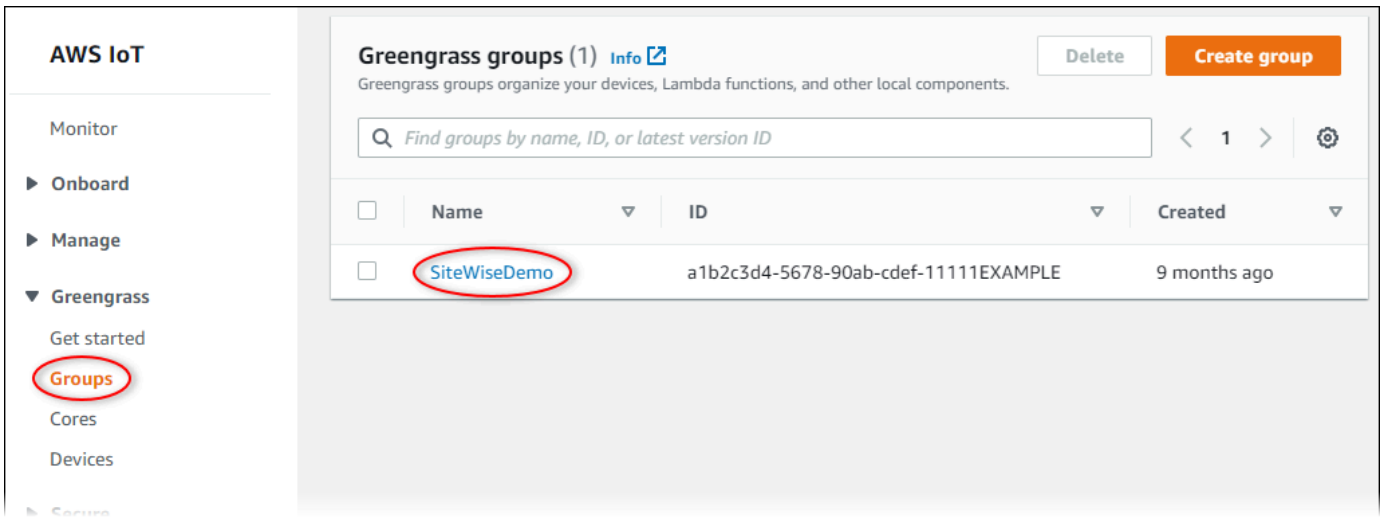
6. Enter a **Description** and choose **Next**.
7. (Optional) On the **Configure automatic rotation** page, configure automatic rotation for your secrets. If you configure automatic rotation, you must redeploy your Greengrass group each time a secret rotates.
8. On the **Configure automatic rotation** page, choose **Next**.
9. Review your new secret and choose **Store**.

Adding secrets to a Greengrass group

In this procedure, you add your source authentication secrets to your AWS IoT Greengrass group to make them available to your IoT SiteWise connector.

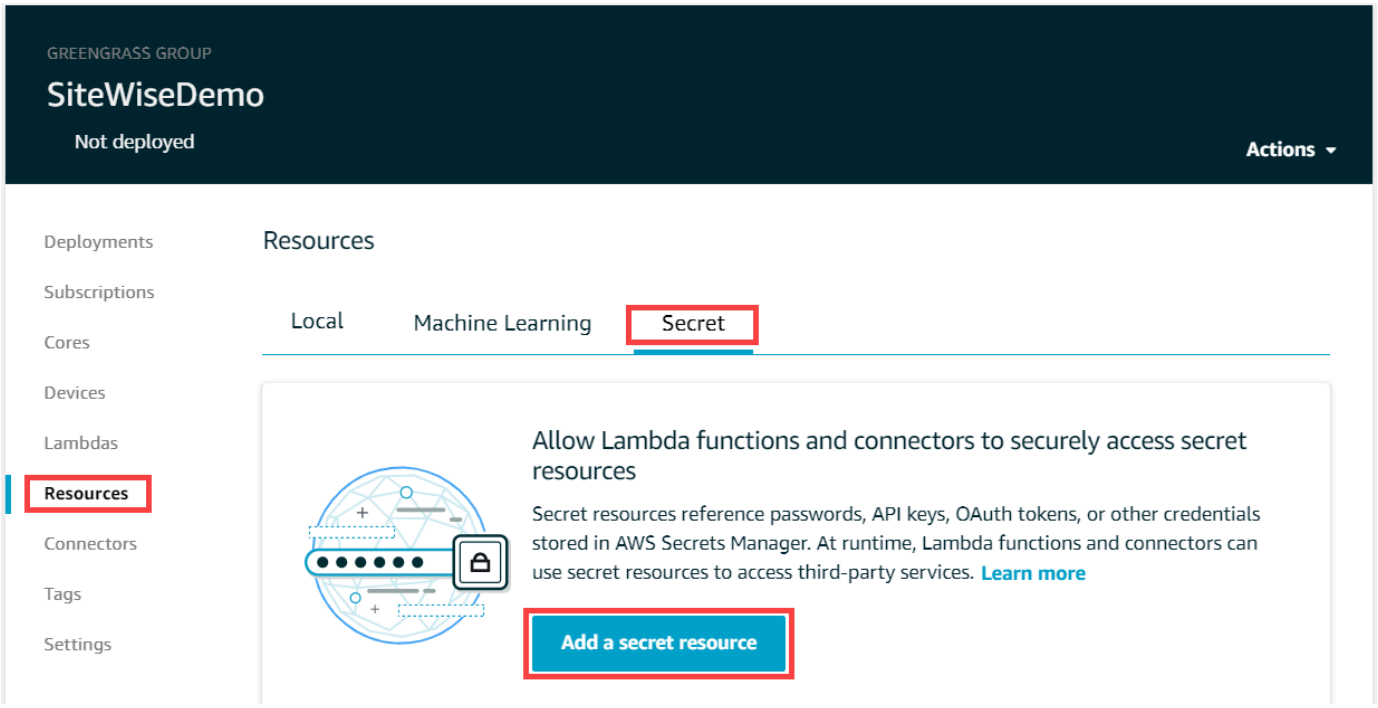
To add a secret to your Greengrass group

1. Navigate to the AWS IoT Greengrass console.
2. In the navigation pane, under **Greengrass**, choose **Groups**, and then choose your group.



3. In the navigation page, choose **Resources**.

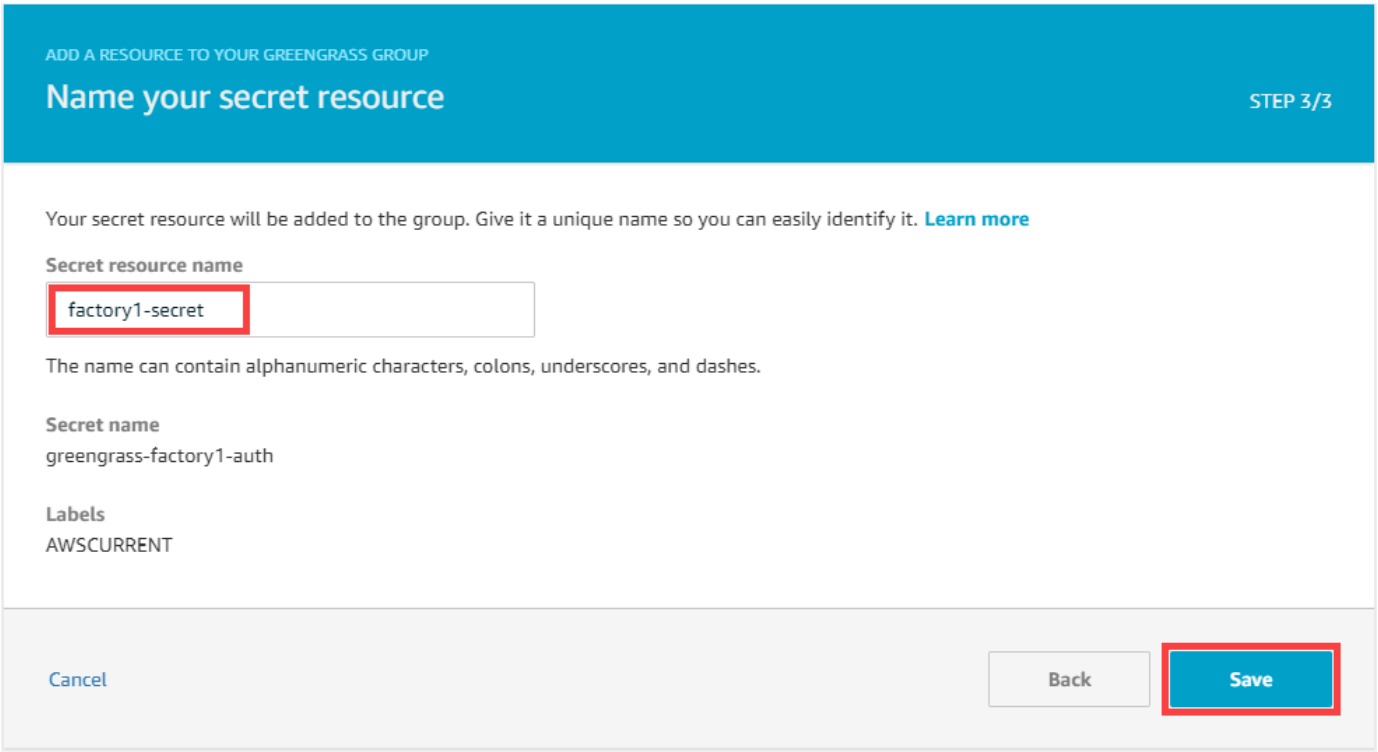
4. On the **Resources** page, choose the **Secret** tab, and then choose **Add a secret resource**.



5. Choose **Select** and choose your secret from the list.

6. Choose **Next**.

7. In **Secret resource name**, enter a name for your secret resource and choose **Save**.

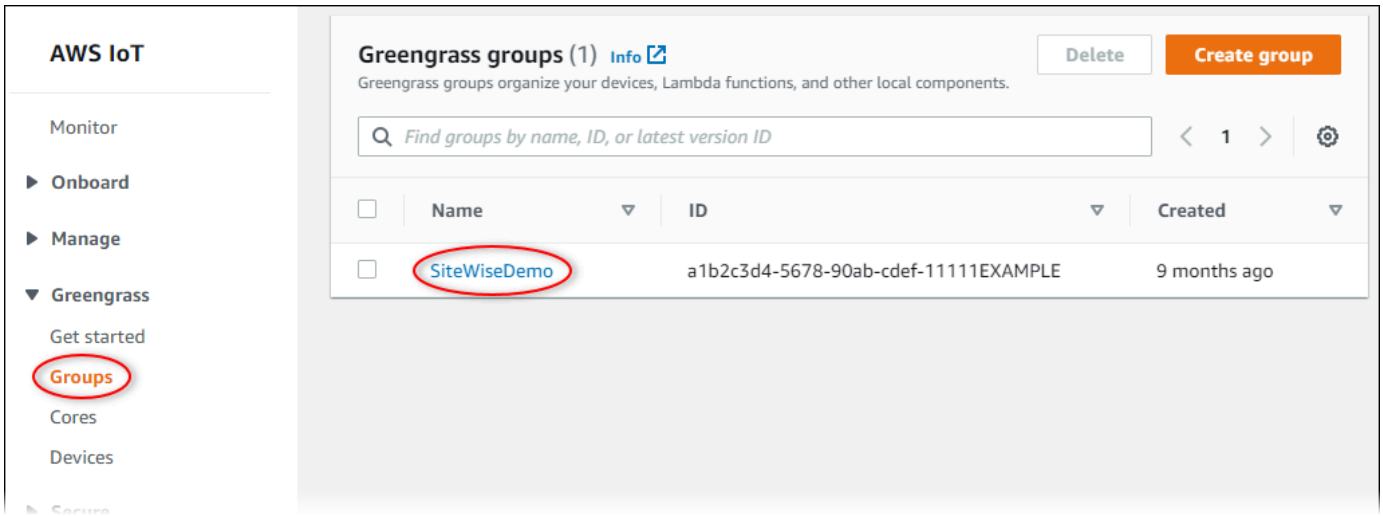


Adding secrets to an IoT SiteWise connector

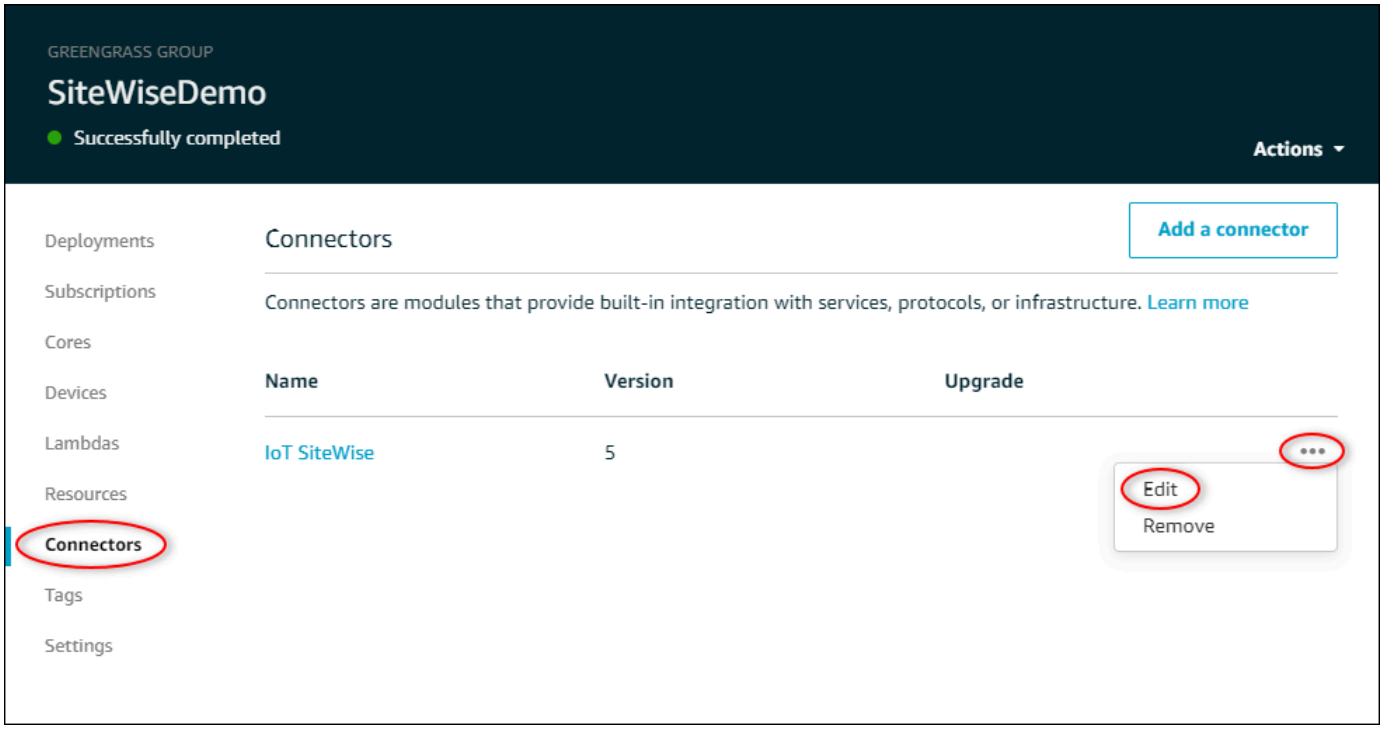
In this procedure, you add your source authentication secrets to your IoT SiteWise connector to make them available to AWS IoT SiteWise and your SiteWise Edge gateway.

To add a secret to your IoT SiteWise connector

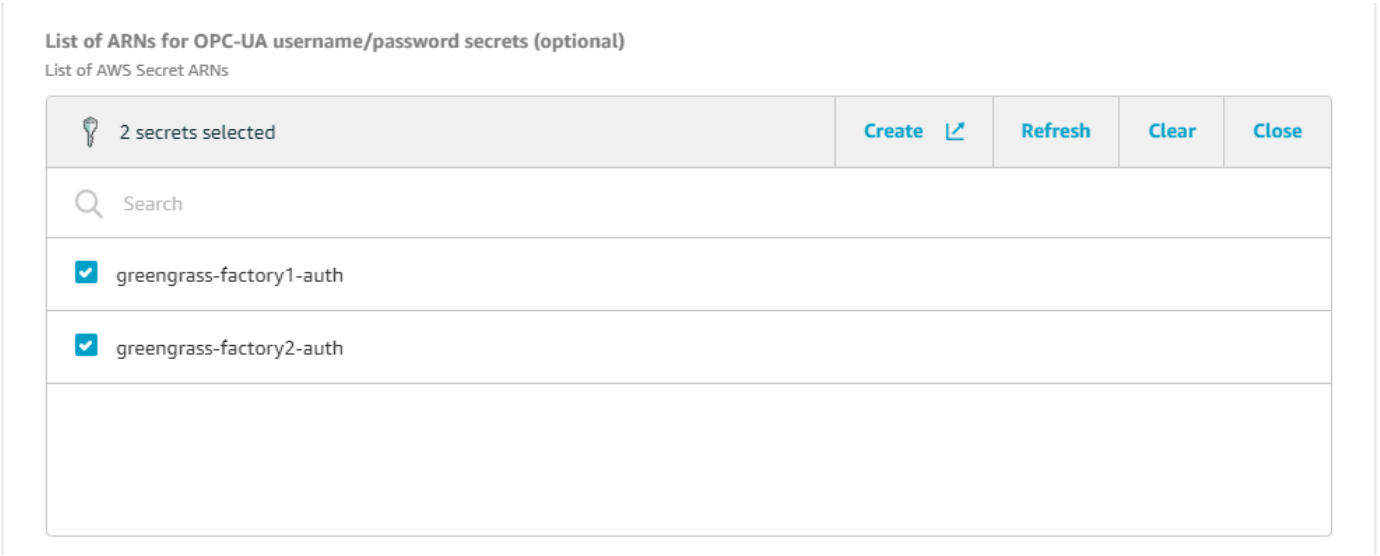
1. Navigate to the AWS IoT Greengrass console.
2. In the navigation pane, under **Greengrass**, choose **Groups**, and then choose your group.



3. In the navigation page, choose **Connectors**.
4. Choose the ellipsis icon for the **IoT SiteWise** connector to open the options menu, and then choose **Edit**.



5. Under **List of ARNs for OPC-UA username/password secrets**, choose **Select**, and then select each secret to add to this SiteWise Edge gateway. If you need to create secrets, see [Creating source authentication secrets].



   If your secret doesn't appear, choose **Refresh**. If your secret still doesn't appear, check that you [added the secret to your Greengrass group].

6. Choose **Save**.

7. In the upper-right corner, in the **Actions** menu, choose **Deploy**.

8. Choose **Automatic detection** to start the deployment.

   If the deployment fails, choose **Deploy** again. If the deployment continues to fail, see AWS IoT Greengrass deployment troubleshooting.

   After your group deploys, you can configure a source that uses the new secret. For more information, see Configuring data sources.

Upgrading a connector

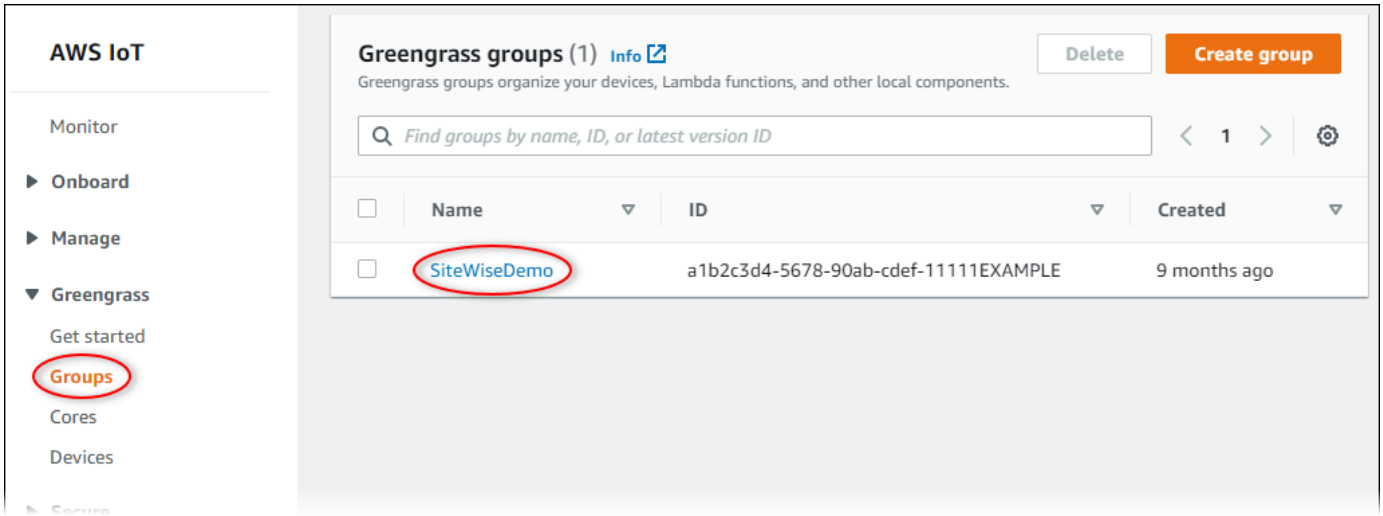
> **Important**
>
> Version 6 of the IoT SiteWise connector introduces new requirements: AWS IoT Greengrass Core software v1.10.0 and stream manager. Before you upgrade your connector, check that your SiteWise Edge gateway meets these requirements, or you won't be able to deploy your SiteWise Edge gateway.

You can easily upgrade your SiteWise Edge gateway's connector after a new IoT SiteWise connector version is released.

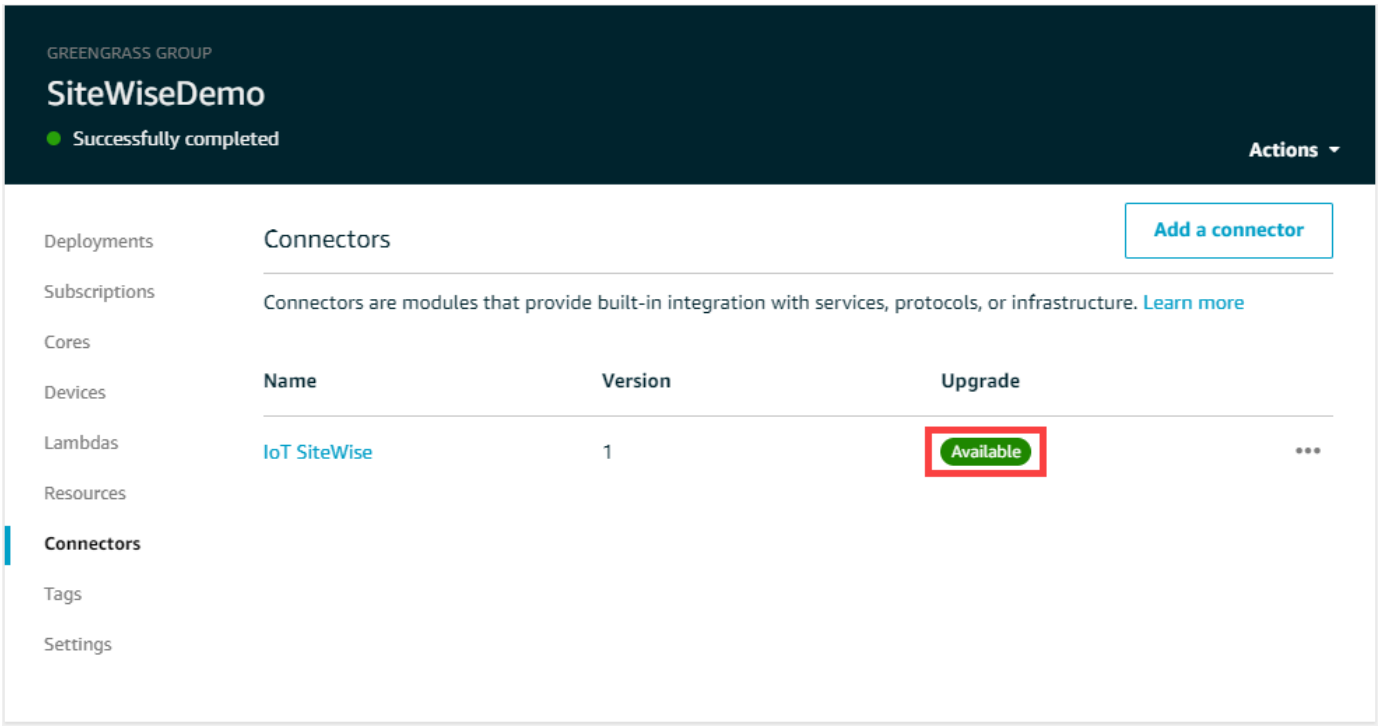
> **Note**
>
> In this procedure, you redeploy your Greengrass group and restart your SiteWise Edge gateway. Your SiteWise Edge gateway won't ingest data while it's restarting. The time to restart your SiteWise Edge gateway depends on the number of tags on your SiteWise Edge gateway's sources. Restart time can range from a few seconds (for a SiteWise Edge gateway with few tags) to several minutes (for a SiteWise Edge gateway with many tags).

To upgrade an IoT SiteWise connector

1. Navigate to the AWS IoT Greengrass console.
2. In the navigation pane, under **Greengrass**, choose **Groups**, and then choose the group that you created when you set up your SiteWise Edge gateway.



3. In the navigation pane, choose **Connectors**.
4. On the **Connectors** page, choose **Available** next to the **IoT SiteWise** connector.



If you don't see the **Available** element, your connector is already the latest version.

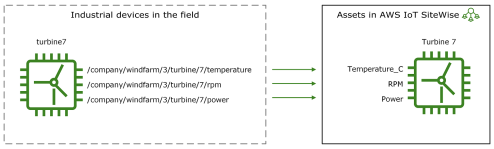
5. On the **Upgrade connector** page, enter your connector's parameters and then choose **Upgrade**.
6. In the upper-right corner, in the **Actions** menu, choose **Deploy**.
7. Choose **Automatic detection** to start the deployment.

If the deployment fails, choose **Deploy** again. If the deployment continues to fail, see AWS IoT Greengrass deployment troubleshooting.

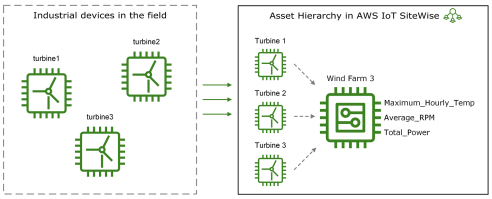
Modeling industrial assets

You can create virtual representations of your industrial operation with AWS IoT SiteWise assets. An **asset** represents a device, a piece of equipment, or a process that uploads one or more data streams to the AWS Cloud. For example, an asset device can be a wind turbine that sends air temperature, propeller rotation speed, and power output time-series measurements to asset properties in AWS IoT SiteWise.

Each data stream corresponds to unique property alias. For example, the alias `/company/windfarm/3/turbine/7/temperature` uniquely identifies the temperature data stream coming from turbine #7 in wind farm #3. You can configure AWS IoT SiteWise assets to transform incoming measurement data using mathematical expressions, such as to convert temperature data from Celsius to Fahrenheit.



An asset can also represent a logical grouping of devices, such as an entire wind farm. You can associate assets with other assets to create asset hierarchies that represent complex industrial operations. Assets can access the data within their associated child assets. By doing so, you can use AWS IoT SiteWise expressions to calculate aggregate metrics, such as the net power output of a wind farm.



You must create every asset from an **asset model**. Asset models are declarative structures that standardize the format of your assets. Asset models enforce consistent information across multiple assets of the same type so that you can process data in assets that represent groups of devices. In the preceding diagram, you use the same asset model for all three turbines because all turbines share a common set of properties.

You can also create **component models**. A component model is a special type of asset model that you can include in asset models or other component models. You can use component models to

define common reusable sub-assemblies, such as sensors, motors, and so forth, that you share across multiple asset models.

After you define your asset models, you can create your industrial assets. To create an asset, select an `ACTIVE` asset model to create an asset from that model. Then, you can populate asset-specific information such as data stream aliases and attributes. In the preceding diagram, you create three turbine assets from one asset model and then associate data stream aliases like `/company/windfarm/3/turbine/7/temperature` for each turbine.

You can also update and delete existing assets, asset models, and component models. When you update an asset model, every asset based on that asset model reflects any changes that you make to the underlying model. When you update a component model, this applies to every asset based on every asset model that references the component model.

Your asset models may be very complex, for example when modeling a complicated piece of equipment that has many subcomponents. To help keep such asset models organized and maintainable, you can use custom composite models to group related properties or to re-use shared components. For more information, see Custom composite models (Components).

Topics

Asset and model states

When you create, update, or delete an asset, an asset model, or a component model, the changes take time to propagate. AWS IoT SiteWise resolves these operations asynchronously and updates the status of each resource. Each asset, asset model, and component model has a status field that contains the state of the resource and any error message, if applicable. The state can be one of the following values:

- `ACTIVE` – The resource is active. This is the only state in which you can query and interact with assets, asset models, and component models.
- `CREATING` – The resource is being created.
- `UPDATING` – The resource is being updated.
- `DELETING` – The resource is being deleted.
- `PROPAGATING` – (Asset models and component models only) The changes are propagating to all dependent resources (from asset model to assets, or from component model to asset models).
- `FAILED` – The resource failed to validate during a create or update operation, possibly due to a circular reference in an expression. You can delete resources that are in the `FAILED` state.

Some of the create, update, and delete operations in AWS IoT SiteWise place an asset, asset model, or component model in a state other than `ACTIVE` while the operation resolves. To query or interact with a resource after you perform one of these operations, you must wait until the state changes to `ACTIVE`. Otherwise, your requests fail.

Topics

Checking the status of an asset

You can use the AWS IoT SiteWise console or API to check the status of an asset.

Topics

Checking the status of an asset (console)

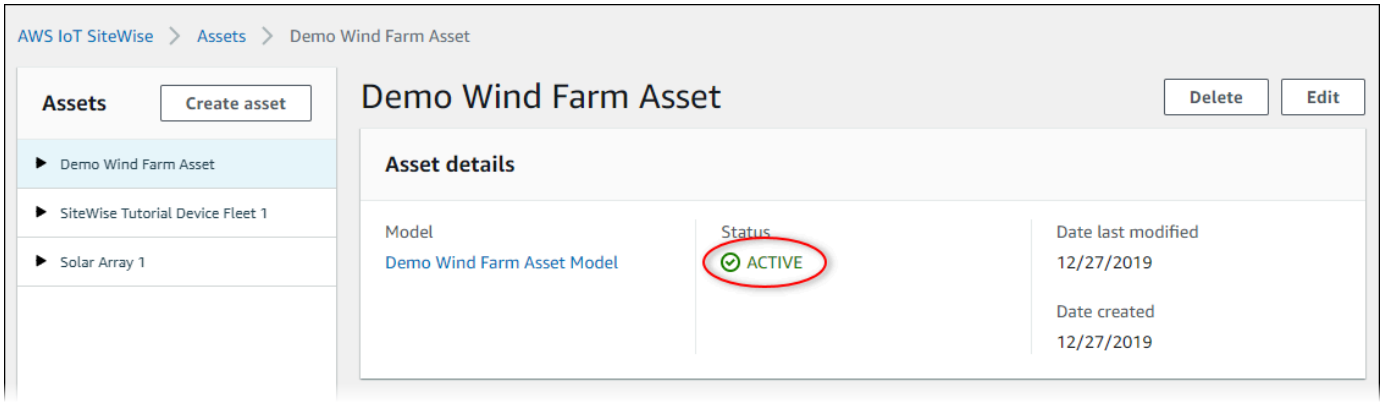
Use the following procedure to check the status of an asset in the AWS IoT SiteWise console.

To check the status of an asset (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the asset to check.

> **Tip**
>
> You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. Find **Status** in the **Asset details** panel.



Checking the status of an asset (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to check the status of an asset.

To check the status of an asset, use the DescribeAsset operation with the `assetId` parameter.

To check the status of an asset (AWS CLI)

- Run the following command to describe the asset. Replace *`asset-id`* with the asset's ID or external ID. The external ID is a user-defined ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

```
aws iotsitewise describe-asset --asset-id asset-id
```

The operation returns a response that contains the asset's details. The response contains an `assetStatus` object that has the following structure:

```
{
    ...
    "assetStatus": {
      "state": "String",
      "error": {
         "code": "String",
         "message": "String"
      }
    }
  }
```

The asset's state is in `assetStatus.state` in the JSON object.

Checking the status of an asset model or component model

You can use the AWS IoT SiteWise console or API to check the status of an asset model or component model.

Topics

- Checking the status of an asset model or component model (console)
- Checking the status of an asset model or component model (AWS CLI)

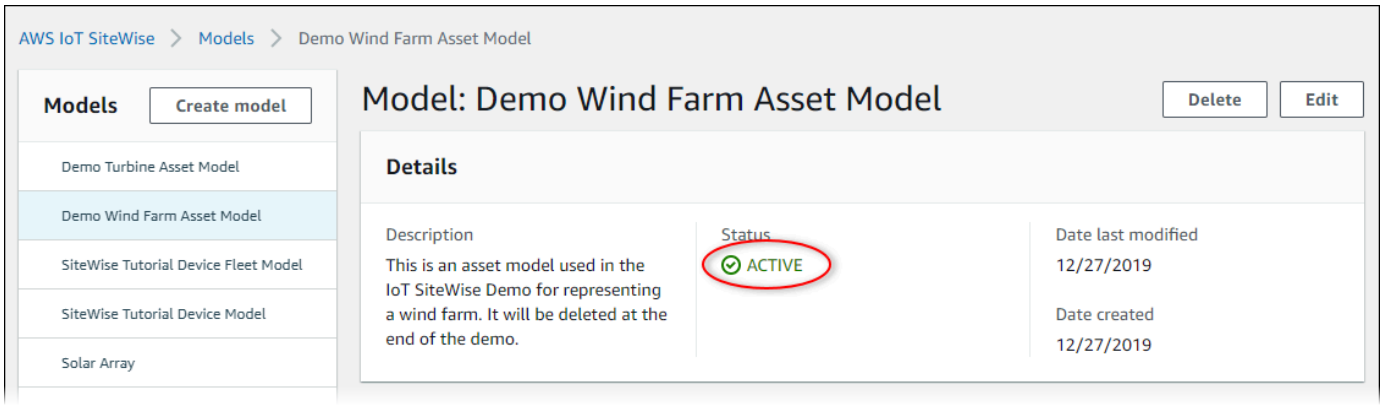
Checking the status of an asset model or component model (console)

Use the following procedure to check the status of an asset model or component model in the AWS IoT SiteWise console.

> **Tip**
>
> Asset models and component models are both listed under **Models** in the navigation pane. The **Details** panel of the selected asset model or component model indicates which type it is.

To check the status of an asset model or component model (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Models**.
3. Choose the model to check.
4. Find **Status** in the **Details** panel.



Checking the status of an asset model or component model (AWS CLI)

You can use the AWS CLI to check the status of an asset model or component model.

To check the status of an asset model or component model, use the DescribeAssetModel operation with the `assetModelId` parameter.

> **Tip**
>
> The AWS CLI defines component models as a type of asset model. Therefore, you use the same DescribeAssetModel operation for both types of model. The `assetModelType` field in the response indicates whether it's an `ASSET_MODEL` or a `COMPONENT_MODEL`.

To check the status of an asset model or component model (AWS CLI)

- Run the following command to describe the model. Replace *`asset-model-id`* with the ID or the external ID of the asset model or component model. The external ID is a user-defined ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id
```

The operation returns a response that contains the model's details. The response contains an `assetModelStatus` object that has the following structure.

```
{
    ...
    "assetModelStatus": {
      "state": "String",
      "error": {
         "code": "String",
         "message": "String"
      }
    }
  }
```

The model's state is in `assetModelStatus.state` in the JSON object.

Custom composite models (Components)

When you're modeling an especially complex industrial asset, such as a complicated piece of machinery that has many parts, it can become a challenge to keep your asset models organized and maintainable.

In such cases, you can add custom composite models, or components if you're using the console, to your existing asset models and component models. These help you stay organized by grouping related properties and re-using subcomponent definitions.

There are two types of custom composite models:

- **Inline** custom composite models define a set of grouped properties that apply to the asset model or component model to which the custom composite model belongs. You use them to group related properties. They consists of a name, a description, and a set of asset model properties. They are not reusable.
- **Component-model-based** custom composite models reference a component model that you want to include in your asset model or component model. You use them to include standard subassemblies in your model. They consist of a name, a description, and the ID of the component

model it references. They have no properties of their own; the referenced component model provides its associated properties to any created assets.

The following sections illustrate how to use custom composite models in your designs.

Topics

Inline custom composite models

Inline custom composite models provide a way to organize your asset model by grouping related properties.

For example, suppose you want to model a robot asset. The robot includes a servomotor, a power supply, and a battery. Each of those constituent parts has its own properties that you want to include in the model. You might define an asset model called `robot_model` that has properties such as the following.

- `robot_model`
  - `servo_status` *(integer)*
  - `servo_position` *(double)*
  - `powersupply_status` *(integer)*
  - `powersupply_temperature` *(double)*
  - `battery_status` *(integer)*
  - `battery_charge` *(double)*

However, in some cases, there might be many subassemblies, or the subassemblies themselves might have many properties. In these cases, there might be so many properties that they become cumbersome to reference and maintain in a single flat list at the model root, like in the preceding example.

To deal with such situations, you can use an inline custom composite model to group properties. An inline custom composite model is a custom composite model that defines its own properties. For example, you could model your robot like the following.

- `robot_model`
  - `servo`
    - `status` *(integer)*
    - `position` *(double)*
  - `powersupply`
    - `status` *(integer)*
    - `temperature` *(double)*
  - `battery`
    - `status` *(integer)*
    - `charge` *(double)*

In the preceding example, `servo`, `powersupply`, and `battery` are the names of inline custom composite models defined within the `robot_model` asset model. Each of these composite models then defines properties of its own.

> **Note**
>
> In this case, each custom composite model defines its own properties, so that all the properties are part of the asset model itself (`robot_model` in this case). These properties aren't shared with any other asset models or component models. For example, if you created some other asset model that also had an inline custom composite model called `servo`, then making a change to the `servo` within `robot_model` wouldn't affect the other asset model's `servo` definition.
>
> If you want to implement such sharing (for example, to have only one definition for a servo, which all your asset models can share), you would create a component model for it instead, and then create **component-model-based** composite models that reference it. See the following section for details.

For information about how to create inline custom composite models, see Creating custom composite models (Components).

Component-model-based custom composite models

You can create a component model in AWS IoT SiteWise to define a standard, reusable sub-assembly. Once you have created a component model, you can add references to it in your other asset models and component models. You do this by adding a **component-model-based custom composite model** to any model where you want to reference the component. You can add references to your component from many models, or multiple times within the same model.

In this way, you can avoid duplicating the same definitions across models. It also simplifies maintaining your models, because any changes you make to a component model will be reflected across all asset models that use it.

For example, suppose that your industrial installation has many types of equipment that all use the same kind of servo motor. Some of them have many servo motors in a single piece of equipment. You create an asset model for each equipment type, but you don't want to duplicate the definition of `servo` every time. You want to model it just once and use it in your various asset models. If you later make a change to the definition of `servo`, it will be updated across all your models and assets.

To model the robot from the previous example in this way, you could define servo motors, power supplies, and batteries as component models, like this.

- `servo_component_model`
  - `status` *(integer)*
  - `position` *(double)*

- `powersupply_component_model`
  - `status` *(integer)*
  - `temperature` *(double)*

- `battery__component_model`

- `status` *(integer)*
- `charge` *(double)*

You could then define asset models, such as `robot_model`, that reference these components. Multiple asset models can reference the same component model. You can also reference the same component model multiple times in one asset model, such as if your robot has multiple servomotors in it.

- `robot_model`
  - `servo1` (*reference:* `servo_component_model`)
  - `servo2` (*reference:* `servo_component_model`)
  - `servo3` (*reference:* `servo_component_model`)
  - `powersupply` (*reference:* `powersupply_component_model`)
  - `battery` (*reference:* `battery_component_model`)

For information about how to create component models, see Creating component models.

For information about how to reference your component models in other models, see Creating custom composite models (Components).

Using paths to reference custom composite model properties

When you create a property on an asset model, component model, or custom composite model, you can reference it from other properties that use its value, such as transforms and metrics.

AWS IoT SiteWise provides different ways for you to reference your property. The simplest way is often to use its property ID. However, if the property you want to reference is on a custom composite model, you may find it more useful to reference it by *path* instead.

A path is an ordered sequence of *path segments* that specifies a property in terms of its position among the nested composite models within an asset model and composite model.

Obtaining property paths

You can get a property's path from the `path` field of its AssetModelProperty.

For example, suppose you have an asset model `robot_model` that contains a custom composite model `servo`, which has a property `position`. If you call DescribeAssetModelCompositeModel on `servo`, then the `position` property would list a `path` field that looks like this:

```
"path": [
    {
       "id": "asset model ID",
       "name": "robot_model"
    },
    {
       "id": "composite model ID",
       "name": "servo"
    },
    {
       "id": "property ID",
       "name": "position"
    }
]
```

Using property paths

You can use a property path when you define a property that references other properties, such as a transform or metric.

A property uses a *variable* to reference another property. For more information about working with variables, see Using variables in formula expressions.

When you define a variable to reference a property, you can use either the property's ID or its path.

To define a variable that uses the path of the referenced property, specify the `propertyPath` field of its value.

For example, to define an asset model that has a metric that references a property by using a path, you could pass a payload like this to CreateAssetModel:

```
{
    ...
    "assetModelProperties": [
        {
            ...
            "type": {
```

```
                "metric": {
                    ...
                    "variables": [
                        {
                            "name": "variable name",
                            "value": {
                                "propertyPath": [
                                    path segments
                                ]
                            }
                        }
                    ],
                    ...
                }
            },
            ...
        },
        ...
    ],
    ...
}
```

Working with object IDs

AWS IoT SiteWise defines various types of persistent objects, such as assets, asset models, properties, and hierarchies. All such objects have unique identifiers that you can use to retrieve, update, and delete them.

AWS IoT SiteWise has different options for customers for ID creation. AWS IoT SiteWise generates one for you by default at object creation time. Users can also provide their own IDs to your objects.

Topics

Working with object UUIDs

Every persistent object in AWS IoT SiteWise has a UUID to identify it. For example, asset models have an asset model ID, assets have an asset ID, and so on. This ID is assigned at the time that you create the object, and remains unchanged for the object's lifetime.

When you create a new object, AWS IoT SiteWise generates a unique ID for you by default. You can also provide your own ID at creation time in UUID format.

> **Note**
>
> UUIDs **must** be globally unique within the AWS Region where it's created, and for the same object type. When AWS IoT SiteWise auto-generates an ID for you, it's always unique. If you choose your own ID, make sure that it's unique.

For example, if you create a new asset model by calling CreateAssetModel, you can provide your own UUID in the optional `assetModelId` field of the request.

By contrast, if you omit `assetModelId` from the request, AWS IoT SiteWise generates a UUID for the new asset model.

Using external IDs

To define your own ID in some format other than UUID, you can assign an *external ID*. For example, you can do this if you reuse an ID that you're using in a system that's not AWS, or to be more human-readable. External IDs have a more flexible format. You can use them to reference your objects in AWS IoT SiteWise API operations where you would otherwise use the UUID.

Like the UUIDs, each external ID must be unique within its context. For example, you can't have two asset models with the same external ID. Also, like the UUIDs, an object can only have one external ID in its lifetime, which can't change.

Differences between external IDs and UUIDs

External IDs differ from UUIDs in the following ways:

- Every object has a UUID, but external IDs are optional.
- AWS IoT SiteWise never generates external IDs. You provide these yourself.
- If the object does not already have one, you can assign an external ID at any time.

Format of external IDs

A valid external ID has the following properties:

- Is between 2 and 128 characters long.

- The first and last characters must be alphanumeric (A-Z, a-z, 0-9).
- Characters other than first and last must either be alphanumeric, or else one of the following: `_-.:`

For example, an external ID must conform to the following regular expression:

```
[a-zA-Z0-9][a-zA-Z0-9_\-.:]*[a-zA-Z0-9]+
```

Referencing objects with external IDs

In many places that you could reference an object using its UUID, you can use its external ID instead, if it has one. To do so, append the external ID to the string `externalId:`.

For example, suppose you have an asset model whose UUID (asset model ID) is `a1b2c3d4-5678-90ab-cdef-11111EXAMPLE`, which also has the external ID `myExternalId`. Call DescribeAssetModel to get details about it. You could use either of the following as the value of `assetModelId`:

- With the asset model ID (UUID) itself: `a1b2c3d4-5678-90ab-cdef-11111EXAMPLE`
- With the external ID: `externalId:myExternalId`

```
aws iotsitewise describe-asset-model --asset-model-id a1b2c3d4-5678-90ab-
cdef-11111EXAMPLE
aws iotsitewise describe-asset-model --asset-model-id externalId:myExternalId
```

Note

The `externalId:` prefix is not, itself, part of the external ID. You only need to provide the prefix when you supply an external ID to an API operation that accepts either UUIDs or external IDs. For example, supply the prefix when you query or update an existing object. When you define an external ID for an object, such as when you create an asset model, don't include the prefix.

You can use external IDs in place of UUIDs in this fashion for many API operations in AWS IoT SiteWise, but not all. For example, the GetAssetPropertyValue, **must** use UUIDs; it doesn't support external ID usage.

To determine whether a particular API operation supports this usage, consult the API Reference.

Creating asset models and component models

AWS IoT SiteWise asset models and component models drive standardization of your industrial data. An asset model or component model contains a name, description, asset properties, and (optionally) custom composite models that group properties together, or that reference component models for subassemblies.

- You use an **asset model** to create assets. In addition to the features listed above, an asset model can also contain hierarchy definitions that define relationships among assets.
- A **component model** represents a subassembly within an asset model or another component model. When you create a component model, you can add references to it in asset models and in other component models. However, you can't create assets directly from component models.

After you create an asset model or component model, you can create custom composite models for it to group properties together or to reference existing component models.

For details about how to create asset models and component models, see the following sections.

Topics

Creating asset models

AWS IoT SiteWise asset models drive standardization of your industrial data. An asset model contains a name, description, asset properties, and asset hierarchy definitions. For example, you can define a wind turbine model with temperature, rotations per minute (RPM), and power properties. Then, you can define a wind farm model with a net power output property and a wind turbine hierarchy definition.

Note

- We recommend that you model your operation starting with the lowest-level nodes. For example, create your wind turbine model before you create your wind farm model. Asset hierarchy definitions contain references to existing asset models. With this approach, you can define asset hierarchies as you create your models.
- Asset models can't contain other asset models. If you must define a model that you can reference as a subassembly within another model, you should create a component--> model instead. For more information, see Creating component models.

The following sections describe how to use the AWS IoT SiteWise console or API to create asset models. The following sections also describe the different types of asset properties and asset hierarchies that you can use to create models.

Topics

Creating an asset model (console)

You can use the AWS IoT SiteWise console to create an asset model. The AWS IoT SiteWise console provides various features, such as formula auto completion, that can help you define valid asset models.

To create an asset model (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Models**.
3. Choose **Create model**.
4. On the **Create model** page, do the following:

    a. Enter a **Name** for the asset model, such as **`Wind Turbine`** or **`Wind Turbine Model`**. This name must be unique across all models in your account in this Region.

b. (Optional) Add an **External ID** for the model. This is a user-defined ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

c. (Optional) Add **Measurement definitions** for the model. Measurements represent data streams from your equipment. For more information, see Defining data streams from equipment (measurements).

d. (Optional) Add **Transform definitions** for the model. Transforms are formulas that map data from one form to another. For more information, see Transforming data (transforms).

e. (Optional) Add **Metric definitions** for the model. Metrics are formulas that aggregate data over time intervals. Metrics can input data from associated assets, so that you can calculate values that represent your operation or a subset of your operation. For more information, see Aggregating data from properties and other assets (metrics).

f. (Optional) Add **Hierarchy definitions** for the model. Hierarchies are relationships between assets. For more information, see Defining asset model hierarchies.

g. (Optional) Add tags for the asset model. For more information, see Tagging your AWS IoT SiteWise resources.

h. Choose **Create model**.

When you create an asset model, the AWS IoT SiteWise console navigates to the new model's page. On this page, you can see the model's **Status**, which is initially **CREATING**. This page automatically updates, so you can wait for the model's status to update.

> **Note**
>
> The asset model creation process can take up to a few minutes for complex models. After the asset model status is **ACTIVE**, you can use the asset model to create assets. For more information, see Asset and model states.

5. (Optional) After you create your asset model, you can configure your asset model for the edge. For more information about SiteWise Edge, see Enabling edge data processing.

   a. On the model page, choose **Configure for Edge**.

   b. On the model configuration page, choose the edge configuration for your model. This controls where AWS IoT SiteWise can compute and store properties associated with this asset model. For more information about configuring your model for the edge, see the section called “Setting up edge capability”.

c. For **Custom edge configuration**, choose the location that you want AWS IoT SiteWise to compute and store each of your asset model properties.

> **Note**
>
> Transforms and metrics that are associated must be configured for the same location. For more information about configuring your model for the edge, see the section called "Setting up edge capability".

d. Choose **Save**. On the model page, your **Edge configuration** should now be **Configured**.

Creating an asset model (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to create an asset model.

Use the CreateAssetModel operation to create an asset model with properties and hierarchies. This operation expects a payload with the following structure.

```
{
  "assetModelType": "ASSET_MODEL",
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
  "assetModelHierarchies": Array of AssetModelHierarchyDefinition
}
```

To create an asset model (AWS CLI)

1. Create a file called `asset-model-payload.json` and then copy the following JSON object into the file.

```
{
  "assetModelType": "ASSET_MODEL",
  "assetModelName": "",
  "assetModelDescription": "",
  "assetModelProperties": [

  ],
  "assetModelHierarchies": [
```

```
  ],
  "assetModelCompositeModels": [

  ]
}
```

2. Use your preferred JSON text editor to edit the `asset-model-payload.json` file for the following:

   a. Enter a name (`assetModelName`) for the asset model, such as **`Wind Turbine`** or **`Wind Turbine Model`**. This name must be unique across all asset models and component models in your account in this AWS Region.

   b. (Optional) Enter an external ID (`assetModelExternalId`) for the asset model. This is a user-defined ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

   c. (Optional) Enter a description (`assetModelDescription`) for the asset model, or remove the `assetModelDescription` key-value pair.

   d. (Optional) Define asset properties (`assetModelProperties`) for the model. For more information, see Defining data properties.

   e. (Optional) Define asset hierarchies (`assetModelHierarchies`) for the model. For more information, see Defining asset model hierarchies.

   f. (Optional) Define alarms for the model. Alarms monitor other properties so that you can identify when equipment or processes require attention. Each alarm definition is a composite model (`assetModelCompositeModels`) that standardizes the set of properties that the alarm uses. For more information, see Monitoring data with alarms and Defining alarms on asset models.

   g. (Optional) Add tags (`tags`) for the asset model. For more information, see Tagging your AWS IoT SiteWise resources.

3. Run the following command to create an asset model from the definition in the JSON file.

```
aws iotsitewise create-asset-model --cli-input-json file://asset-model-payload.json
```

   The operation returns a response that contains the `assetModelId` that you refer to when creating an asset. The response also contains the state of the model (`assetModelStatus.state`), which is initially `CREATING`. The asset model's status is `CREATING` until the changes propagate.

> **Note**
>
> The asset model creation process can take up to a few minutes for complex models. To check the current status of your asset model, use the DescribeAssetModel operation by specifying the `assetModelId`. After the asset model status is `ACTIVE`, you can use the asset model to create assets. For more information, see Asset and model states.

4. (Optional) Create custom composite models for your asset model. With custom composite models, you can group properties within the model, or include a subassembly by referencing a component model. For more information, see Creating custom composite models (Components).

Example asset models

This section contains example asset models definitions that you can use to create asset models with the AWS CLI and AWS IoT SiteWise SDKs. These asset models represent a wind turbine and a wind farm. Wind turbine assets ingest raw sensor data and calculate values such as power and average wind speed. Wind farm assets calculate values such as total power for all wind turbines in the wind farm.

Topics

- Wind turbine asset model
- Wind farm asset model

Wind turbine asset model

The following asset model represents a turbine in a wind farm. The wind turbine ingests sensor data to calculate values such as power and average wind speed.

> **Note**
>
> This example model resembles the wind turbine model from the AWS IoT SiteWise demo. For more information, see Using the AWS IoT SiteWise demo.

```
{
```

```
  "assetModelType": "ASSET_MODEL",
  "assetModelName": "Wind Turbine Asset Model",
  "assetModelDescription": "Represents a turbine in a wind farm.",
  "assetModelProperties": [
    {
      "name": "Location",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "Renton"
        }
      }
    },
    {
      "name": "Make",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "Amazon"
        }
      }
    },
    {
      "name": "Model",
      "dataType": "INTEGER",
      "type": {
        "attribute": {
          "defaultValue": "500"
        }
      }
    },
    {
      "name": "Torque (KiloNewton Meter)",
      "dataType": "DOUBLE",
      "unit": "kNm",
      "type": {
        "measurement": {}
      }
    },
    {
      "name": "Wind Direction",
      "dataType": "DOUBLE",
      "unit": "Degrees",
      "type": {
```

```
        "measurement": {}
      }
    },
    {
      "name": "RotationsPerMinute",
      "dataType": "DOUBLE",
      "unit": "RPM",
      "type": {
        "measurement": {}
      }
    },
    {
      "name": "Wind Speed",
      "dataType": "DOUBLE",
      "unit": "m/s",
      "type": {
        "measurement": {}
      }
    },
    {
      "name": "RotationsPerSecond",
      "dataType": "DOUBLE",
      "unit": "RPS",
      "type": {
        "transform": {
          "expression": "rpm / 60",
          "variables": [
            {
              "name": "rpm",
              "value": {
                "propertyId": "RotationsPerMinute"
              }
            }
          ]
        }
      }
    },
    {
      "name": "Overdrive State",
      "dataType": "DOUBLE",
      "type": {
        "transform": {
          "expression": "gte(torque, 3)",
          "variables": [
```

```
          {
            "name": "torque",
            "value": {
              "propertyId": "Torque (KiloNewton Meter)"
            }
          }
        ]
      }
    }
  },
  {
    "name": "Average Power",
    "dataType": "DOUBLE",
    "unit": "Watts",
    "type": {
      "metric": {
        "expression": "avg(torque) * avg(rps) * 2 * 3.14",
        "variables": [
          {
            "name": "torque",
            "value": {
              "propertyId": "Torque (Newton Meter)"
            }
          },
          {
            "name": "rps",
            "value": {
              "propertyId": "RotationsPerSecond"
            }
          }
        ],
        "window": {
          "tumbling": {
            "interval": "5m"
          }
        }
      }
    }
  },
  {
    "name": "Average Wind Speed",
    "dataType": "DOUBLE",
    "unit": "m/s",
    "type": {
```

```
        "metric": {
          "expression": "avg(windspeed)",
          "variables": [
            {
              "name": "windspeed",
              "value": {
                "propertyId": "Wind Speed"
              }
            }
          ],
          "window": {
            "tumbling": {
              "interval": "5m"
            }
          }
        }
      }
    },
    {
      "name": "Torque (Newton Meter)",
      "dataType": "DOUBLE",
      "unit": "Nm",
      "type": {
        "transform": {
          "expression": "knm * 1000",
          "variables": [
            {
              "name": "knm",
              "value": {
                "propertyId": "Torque (KiloNewton Meter)"
              }
            }
          ]
        }
      }
    },
    {
      "name": "Overdrive State Time",
      "dataType": "DOUBLE",
      "unit": "Seconds",
      "type": {
        "metric": {
          "expression": "statetime(overdrive_state)",
          "variables": [
```

```
            {
              "name": "overdrive_state",
              "value": {
                "propertyId": "Overdrive State"
              }
            }
          ],
          "window": {
            "tumbling": {
              "interval": "5m"
            }
          }
        }
      }
    }
  ],
  "assetModelHierarchies": []
}
```

Wind farm asset model

The following asset model represents a wind farm that comprises multiple wind turbines. This asset model defines a hierarchy to the wind turbine model. This lets the wind farm calculate values (such as average power) from data for all wind turbines in the wind farm.

> **Note**
>
> This example model resembles the wind farm model from the AWS IoT SiteWise demo. For more information, see Using the AWS IoT SiteWise demo.

This asset model depends on the Wind turbine asset model. Replace the `propertyId` and `childAssetModelId` values with those from an existing wind turbine asset model.

```
{
  "assetModelName": "Wind Farm Asset Model",
  "assetModelDescription": "Represents a wind farm.",
  "assetModelProperties": [
    {
      "name": "Code",
      "dataType": "INTEGER",
      "type": {
```

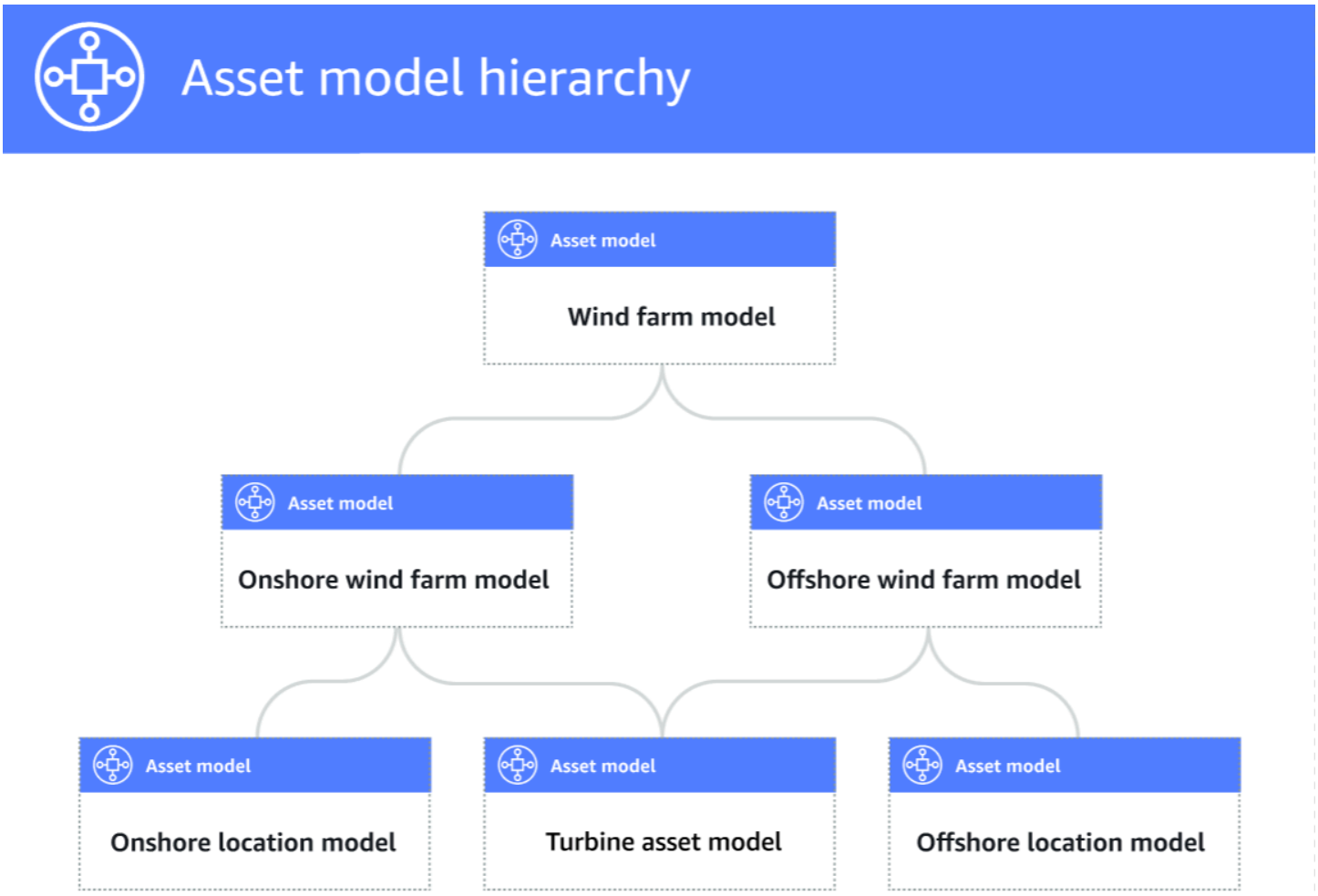
```
        "attribute": {
          "defaultValue": "300"
        }
      }
    },
    {
      "name": "Location",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "Renton"
        }
      }
    },
    {
      "name": "Reliability Manager",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "Mary Major"
        }
      }
    },
    {
      "name": "Total Overdrive State Time",
      "dataType": "DOUBLE",
      "unit": "seconds",
      "type": {
        "metric": {
          "expression": "sum(overdrive_state_time)",
          "variables": [
            {
              "name": "overdrive_state_time",
              "value": {
                "propertyId": "ID of Overdrive State Time property in Wind Turbine
Asset Model",
                "hierarchyId": "Turbine Asset Model"
              }
            }
          ],
          "window": {
            "tumbling": {
              "interval": "5m"
            }
```

```
          }
        }
      }
    },
    {
      "name": "Total Average Power",
      "dataType": "DOUBLE",
      "unit": "Watts",
      "type": {
        "metric": {
          "expression": "sum(turbine_avg_power)",
          "variables": [
            {
              "name": "turbine_avg_power",
              "value": {
                "propertyId": "ID of Average Power property in Wind Turbine Asset
 Model",
                "hierarchyId": "Turbine Asset Model"
              }
            }
          ],
          "window": {
            "tumbling": {
              "interval": "5m"
            }
          }
        }
      }
    }
  ],
  "assetModelHierarchies": [
    {
      "name": "Turbine Asset Model",
      "childAssetModelId": "ID of Wind Turbine Asset Model"
    }
  ]
}
```

Defining asset model hierarchies

You can define asset model hierarchies to create logical associations between the asset models in your industrial operation. For example, you can define a wind farm composed of onshore and

offshore wind farms. An onshore wind farm contains a turbine and onshore location. An offshore wind farm contains a turbine and offshore location.



When you associate a child asset model to a parent asset model through a hierarchy, the parent asset model's metrics can input data from the child asset model's metrics. You can use asset model hierarchies and metrics to calculate statistics that provide insight to your operation or a subset of your operation. For more information, see Aggregating data from properties and other assets (metrics).

Each hierarchy defines a relationship between a parent asset model and a child asset model. In a parent asset model, you can define multiple hierarchies to the same child asset model. For example, if you have two different types of wind turbines in your wind farms, where all wind turbines are represented by the same asset model, you can define a hierarchy for each type. Then, you can define metrics in the wind farm model to calculate independent and combined statistics for each type of wind turbine.

A parent asset model can be associated with multiple child asset models. For example, if you have an onshore wind farm and an offshore wind farm that are represented by two different asset models, you can associate these asset models with the same parent wind farm asset model.

A child asset model can also be associated with multiple parent asset models. For example, if you have two different types of wind farms, where all wind turbines are represented by the same asset model, you can associate the wind turbine asset model with different wind farm asset models.

> **Note**
>
> When you define an asset model hierarchy, the child asset model must be `ACTIVE` or have a previous `ACTIVE` version. For more information, see Asset and model states.

After you define hierarchical asset models and create assets, you can associate the assets to complete the parent-child relationship. For more information, see Creating assets and Associating and disassociating assets.

Topics

- Defining asset model hierarchies (console)
- Defining asset hierarchies (AWS CLI)

Defining asset model hierarchies (console)

When you define a hierarchy for an asset model in the AWS IoT SiteWise console, you specify the following parameters:

- **Hierarchy name** – The hierarchy's name, such as **`Wind Turbines`**.
- **Hierarchy model** – The child asset model.
- **Hierarchy External ID** (Optional) – This is a user-defined ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

For more information, see Creating an asset model (console).

Defining asset hierarchies (AWS CLI)

When you define a hierarchy for an asset model with the AWS IoT SiteWise API, you specify the following parameters:

- `name` – The hierarchy's name, such as **`Wind Turbines`**.
- `childAssetModelId` – The ID or the external ID of the child asset model for the hierarchy. You can use the ListAssetModels operation to find the ID of an existing asset model.

Example Example hierarchy definition

The following example demonstrates an asset model hierarchy that represents a wind farm's relationship to wind turbines. This object is an example of an AssetModelHierarchy. For more information, see Creating an asset model (AWS CLI).

```
{
  ...
  "assetModelHierarchies": [
    {
      "name": "Wind Turbines",
      "childAssetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
    },
  ]
}
```

Creating component models

Use AWS IoT SiteWise component models to define subassemblies that you can reference from asset models or other component models. In this way, you can re-use the definition of the component across multiple other models, or multiple times within the same model.

The process of defining a component model is very similar to defining an asset model. Like an asset model, a component model has a name, description, and asset properties. However, component models can't include asset hierarchy definitions, since component models themselves can't be used to create assets directly. Component models also can't define alarms.

For example, you can define a component for a servo motor with motor temperature, encoder temperature, and insulation resistance properties. Then, you can define an asset model for equipment that contains servo motors, such as a CNC machine.

Note

- We recommend that you model your operation starting with the lowest-level nodes. For example, create your servo motor component before you create your CNC machine's asset model. Asset models contain references to existing component models.
- You can't create an asset directly from a component model. To create an asset that uses your component, you must create an asset model for your asset. Then, you create a custom composite model for it that references your component. For more information about creating asset models, see Creating asset models For more information about creating custom composite models, see Creating custom composite models (Components).

The following sections describe how to use the AWS IoT SiteWise API to create component models.

Topics

Creating a component model (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to create a component model.

Use the CreateAssetModel operation to create a component model with properties. This operation expects a payload with the following structure:

```
{
  "assetModelType": "COMPONENT_MODEL",
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
}
```

To create a component model (AWS CLI)

1. Create a file called `component-model-payload.json` and then copy the following JSON object into the file:

```
{
  "assetModelType": "COMPONENT_MODEL",
  "assetModelName": "",
  "assetModelDescription": "",
  "assetModelProperties": [

  ]
}
```

2. Use your preferred JSON text editor to edit the `component-model-payload.json` file for the following:

   a. Enter a name (`assetModelName`) for the component model, such as **`Servo Motor`** or **`Servo Motor Model`**. This name must be unique across all asset models and component models in your account in this AWS Region.

   b. (Optional) Enter an external ID (`assetModelExternalId`) for the component model. This is a user-defined ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

   c. (Optional) Enter a description (`assetModelDescription`) for the asset model, or remove the `assetModelDescription` key-value pair.

   d. (Optional) Define asset properties (`assetModelProperties`) for the component model. For more information, see Defining data properties.

   e. (Optional) Add tags (`tags`) for the asset model. For more information, see Tagging your AWS IoT SiteWise resources.

3. Run the following command to create a component model from the definition in the JSON file.

```
aws iotsitewise create-asset-model --cli-input-json file://component-model-
payload.json
```

   The operation returns a response that contains the `assetModelId` that you refer to when adding a reference to your component model in an asset model or another component model. The response also contains the state of the model (`assetModelStatus.state`), which is initially `CREATING`. The component model's status is `CREATING` until the changes propagate.

> **Note**
>
> The component model creation process can take up to a few minutes for complex models. To check the current status of your component model, use the DescribeAssetModel operation by specifying the `assetModelId`. After the component model status is `ACTIVE`, you can add references to your component model in asset models or other component models. For more information, see Asset and model states.

4. (Optional) Create custom composite models for your component model. With custom composite models, you can group properties within the model, or to include a subassembly by referencing another component model. For more information, see Creating custom composite models (Components).

Example component model

This section contains an example component model definition that you can use to create a component model with the AWS CLI and AWS IoT SiteWise SDKs. This component model represents a servo motor that can be used within another piece of equipment, such as a CNC machine.

Topics

- Servo motor component model

Servo motor component model

The following component model represents a servo motor that can be used within equipment such as CNC machines. The servo motor provides various measurements, such as temperatures and electrical resistance. These measurements are available as properties on assets created from asset models that reference the servo motor component model.

```
{
    "assetModelName": "ServoMotor",
    "assetModelType": "COMPONENT_MODEL",
    "assetModelProperties": [
        {
            "dataType": "DOUBLE",
            "name": "Servo Motor Temperature",
```

```
            "type": {
            "measurement": {}
            },
            "unit": "Celsius"
        },
        {
            "dataType": "DOUBLE",
            "name": "Spindle speed",
            "type": {
            "measurement": {}
            },
            "unit": "rpm"
        }
    ]
}
```

Defining data properties

Asset properties are the structures within each asset that contain asset data. Asset properties can be any of the following types:

- **Attributes** – An asset's generally static properties, such as device manufacturer or geographic region. For more information, see Defining static data (attributes).
- **Measurements** – An asset's raw device's sensor data streams, such as timestamped rotation speed values or timestamped temperature values in Celsius. A measurement is defined by a data stream alias. For more information, see Defining data streams from equipment (measurements).
- **Transforms** – An asset's transformed time-series values, such as timestamped temperature values in Fahrenheit. A transform is defined by an expression and the variables to consume with that expression. For more information, see Transforming data (transforms).
- **Metrics** – An asset's data aggregated over a specified time interval, such as the hourly average temperature. A metric is defined by a time interval, an expression, and the variables to consume with that expression. Metric expressions can input associated assets' metric properties, so that you can calculate metrics that represent your operation or a subset of your operation. For more information, see Aggregating data from properties and other assets (metrics).

For more information, see Creating asset models.

For an example of how to use measurements, transforms, and metrics to calculate Overall Equipment Effectiveness (OEE), see Calculating OEE in AWS IoT SiteWise.

Topics

Defining static data (attributes)

Asset attributes represent information that is generally static, such as device manufacturer or geographic location. Each asset that you create from an asset model contains the attributes of that model.

Topics

Defining attributes (console)

When you define an attribute for an asset model in the AWS IoT SiteWise console, you specify the following parameters:

- **Name** – The property's name.
- **Default value** – (Optional) The default value for this attribute. Assets created from the model have this value for the attribute. For more information about how to override the default value in an asset created from a model, see Updating attribute values.
- **Data type** – The property's data type, which is one of the following:
  - **String** – A string with up to 1024 bytes.
  - **Integer** – A signed 32-bit integer with range [-2,147,483,648, 2,147,483,647].
  - **Double** – A floating point number with range [-10^100, 10^100] and IEEE 754 double precision.
  - **Boolean** – `true` or `false`.
- **External ID** – (Optional) This is a user-defined ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

For more information, see Creating an asset model (console).

Defining attributes (AWS CLI)

When you define an attribute for an asset model with the AWS IoT SiteWise API, you specify the following parameters:

- `name` – The property's name.
- `defaultValue` – (Optional) The default value for this attribute. Assets created from the model have this value for the attribute. For more information about how to override the default value in an asset created from a model, see Updating attribute values.
- `dataType` – The property's data type, which is one of the following:
  - `STRING` – A string with up to 1024 bytes.
  - `INTEGER` – A signed 32-bit integer with range [-2,147,483,648, 2,147,483,647].
  - `DOUBLE` – A floating point number with range [-10^100, 10^100] and IEEE 754 double precision.
  - `BOOLEAN` – `true` or `false`.
- `externalId` – (Optional) This is a user-defined ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

Example Example attribute definition

The following example demonstrates an attribute that represents an asset's model number with a default value. This object is an example of an AssetModelProperty that contains an Attribute. You can specify this object as a part of the CreateAssetModel request payload to create an attribute property. For more information, see Creating an asset model (AWS CLI).

```
{
...
"assetModelProperties": [
{
  "name": "Model number",
  "dataType": "STRING",
  "type": {
    "attribute": {
      "defaultValue": "BLT123"
    }
  }
```

```
}
],
...
}
```

Defining data streams from equipment (measurements)

A *measurement* represents a device's raw sensor data stream, such as timestamped temperature values or timestamped rotations per minute (RPM) values.

Topics

- Defining measurements (console)
- Defining measurements (AWS CLI)

Defining measurements (console)

When you define a measurement for an asset model in the AWS IoT SiteWise console, you specify following parameters:

- **Name** – The property's name.
- **Unit** – (Optional) The scientific unit for the property, such as mm or Celsius.
- **Data type** – The property's data type, which is one of the following:
  - **String** – A string with up to 1024 bytes.
  - **Integer** – A signed 32-bit integer with range [-2,147,483,648, 2,147,483,647].
  - **Double** – A floating point number with range [-10^100, 10^100] and IEEE 754 double precision.
  - **Boolean** – `true` or `false`.
- **External ID** – (Optional) This is a user-defined ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

For more information, see Creating an asset model (console).

Defining measurements (AWS CLI)

When you define a measurement for an asset model with the AWS IoT SiteWise API, you specify the following parameters:

- `name` – The property's name.
- `dataType` – The property's data type, which is one of the following:
  - `STRING` – A string with up to 1024 bytes.
  - `INTEGER` – A signed 32-bit integer with range [-2,147,483,648, 2,147,483,647].
  - `DOUBLE` – A floating point number with range [-10^100, 10^100] and IEEE 754 double precision.
  - `BOOLEAN` – `true` or `false`.
- `unit` – (Optional) The scientific unit for the property, such as mm or Celsius.
- `externalId` – (Optional) This is a user-defined ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

Example Example measurement definition

The following example demonstrates a measurement that represents an asset's temperature sensor readings. This object is an example of an AssetModelProperty that contains a Measurement. You can specify this object as a part of the CreateAssetModel request payload to create a measurement property. For more information, see Creating an asset model (AWS CLI).

The Measurement structure is an empty structure when you define an asset model because you later configure each asset to use unique device data streams. For more information about how to connect an asset's measurement property to a device's sensor data stream, see Mapping industrial data streams to asset properties.

```
{
      ...
      "assetModelProperties": [
      {
          "name": "Temperature C",
          "dataType": "DOUBLE",
          "type": {
              "measurement": {}
          },
          "unit": "Celsius"
      }
  ],
      ...
}
```

Transforming data (transforms)

Transforms are mathematical expressions that map asset properties' data points from one form to another. A transform expression consists of asset property variables, literals, operators, and functions. The transformed data points hold a one-to-one relationship with the input data points. AWS IoT SiteWise calculates a new transformed data point each time any of the input properties receives a new data point.

For example, if your asset has a temperature measurement stream named `Temperature_C` with units in Celsius, you can convert each data point to Fahrenheit with the formula `Temperature_F = 9/5 * Temperature_C + 32`. Each time AWS IoT SiteWise receives a data point in the `Temperature_C` measurement stream, the corresponding `Temperature_F` value is calculated within a few seconds and available as the `Temperature_F` property.

If your transform contains more than one variable, the data point that arrives earlier initiates the computation immediately. Consider an example where a parts manufacturer uses a transform to monitor product quality. Using a different standard based on the part type, the manufacturer uses the following measurements to represent the process:

- `Part_Number` - A string that identifies the part type.
- `Good_Count` - An integer that increases by one if the part meets the standard.
- `Bad_Count` - An integer that increases by one if the part doesn't meet the standard.

The manufacturer also creates a transform, `Quality_Monitor`, that equals `if(eq(Part_Number, "BLT123") and (Bad_Count / (Good_Count + Bad_Count) > 0.1), "Caution", "Normal")`.

This transform monitors the percentage of bad parts produced for a specific part type. If the part number is BLT123 and the percentage of bad parts exceeds 10 percent (0.1), the transform returns `"Caution"`. Otherwise, the transform returns `"Normal"`.

Note

- If `Part_Number` receives a new data point before other measurements, the `Quality_Monitor` transform uses the new `Part_Number` value and the latest `Good_Count` and `Bad_Count` values. To avoid errors, reset `Good_Count` and `Bad_Count` before the next manufacturing run.

- Use metrics if you want to evaluate expressions only after all variables receive new data points.

Topics

- Defining transforms (console)
- Defining transforms (AWS CLI)

Defining transforms (console)

When you define a transform for an asset model in the AWS IoT SiteWise console, you specify following parameters:

- **Name** – The property's name.
- **Unit** – (Optional) The scientific unit for the property, such as mm or Celsius.
- **Data type** – The data type of the transform, which can be **Double** or **String**.
- **External ID** – (Optional) This is a user-defined ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.
- **Formula** – The transform expression. Transform expressions can't use aggregation functions or temporal functions. To open the auto complete feature, start typing or press the down arrow key. For more information, see Using formula expressions.

> **Important**
>
> Transforms can input properties that are integer, double, Boolean, or string type. Booleans convert to `0` (false) and `1` (true).
> Transforms must input one or more properties that aren't attributes and any number of attribute properties. AWS IoT SiteWise calculates a new transformed data point each time the input property that isn't an attribute receives a new data point. New attribute values don't launch transform updates. The same request rate for asset property data API operations applies for transform computation results.
> Formula expressions can only output double or string values. Nested expressions can output other data types, such as strings, but the formula as a whole must evaluate to a number or string. You can use the jp function to convert a string to a number. The

Boolean value must be 1 (true) or 0 (false). For more information, see Undefined, infinite, and overflow values.

For more information, see Creating an asset model (console).

Defining transforms (AWS CLI)

When you define a transform for an asset model with the AWS IoT SiteWise API, you specify the following parameters:

- `name` – The property's name.
- `unit` – (Optional) The scientific unit for the property, such as mm or Celsius.
- `dataType` – The data type of the transform, which must be `DOUBLE` or `STRING`.
- `externalId` – (Optional) This is a user-defined ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.
- `expression` – The transform expression. Transform expressions can't use aggregation functions or temporal functions. For more information, see Using formula expressions.
- `variables` – The list of variables that defines the other properties of your asset to use in the expression. Each variable structure contains a simple name to use in the expression and a `value` structure that identifies which property to link to that variable. The `value` structure contains the following information:
  - `propertyId` – The ID of the property from which to input values. You can use the property's name instead of its ID.

> **Important**
>
> Transforms can input properties that are integer, double, Boolean, or string type. Booleans convert to `0` (false) and `1` (true).
> Transforms must input one or more properties that aren't attributes and any number of attribute properties. AWS IoT SiteWise calculates a new transformed data point each time the input property that isn't an attribute receives a new data point. New attribute values don't launch transform updates. The same request rate for asset property data API operations applies for transform computation results.
> Formula expressions can only output double or string values. Nested expressions can output other data types, such as strings, but the formula as a whole must evaluate to a number or string. You can use the jp function to convert a string to a number. The

Boolean value must be 1 (true) or 0 (false). For more information, see Undefined, infinite, and overflow values.

Example transform definition

The following example demonstrates a transform property that converts an asset's temperature measurement data from Celsius to Fahrenheit. This object is an example of an AssetModelProperty that contains a Transform. You can specify this object as a part of the CreateAssetModel request payload to create a transform property. For more information, see Creating an asset model (AWS CLI).

```
{
...
"assetModelProperties": [
...
{
  "name": "Temperature F",
  "dataType": "DOUBLE",
  "type": {
    "transform": {
      "expression": "9/5 * temp_c + 32",
      "variables": [
        {
          "name": "temp_c",
          "value": {
            "propertyId": "Temperature C"
          }
        }
      ]
    }
  },
  "unit": "Fahrenheit"
}
],
...
}
```

Example transform definition that contains three variables

The following example demonstrates a transform property that returns a warning message (`"Caution"`) if more than 10 percent of the BLT123 parts don't meet the standard. Otherwise, it returns an information message (`"Normal"`).

```
{
...
"assetModelProperties": [
...
{
"name": "Quality_Monitor",
"dataType": "STRING",
"type": {
    "transform": {
        "expression": "if(eq(Part_Number,"BLT123") and (Bad_Count / (Good_Count +
 Bad_Count) > 0.1), "Caution", "Normal")",
        "variables": [
            {
                "name": "Part_Number",
                "value": {
                    "propertyId": "Part Number"
                }
            },
            {
                "name": "Good_Count",
                "value": {
                    "propertyId": "Good Count"
                }
            },
            {
                "name": "Bad_Count",
                "value": {
                    "propertyId": "Bad Count"
                }
            }
        ]
    }
}
}
...
}
```

Aggregating data from properties and other assets (metrics)

Metrics are mathematical expressions that use aggregation functions to process all input data points and output a single data point per specified time interval. For example, a metric can calculate the average hourly temperature from a temperature data stream.

Metrics can input data from associated assets' metrics, so you can calculate statistics that provide insight to your operation or a subset of your operation. For example, a metric can calculate the average hourly temperature across all wind turbines in a wind farm. For more information about how to define associations between assets, see Defining asset model hierarchies.

Metrics can also input data from other properties without aggregating data over each time interval. If you specify an attribute in a formula, AWS IoT SiteWise uses the latest value for that attribute when it computes the formula. If you specify a metric in a formula, AWS IoT SiteWise uses the last value for the time interval over which it computes the formula. This means you can define metrics like `OEE = Availability * Quality * Performance`, where `Availability`, `Quality`, and `Performance` are all other metrics on the same asset model.

AWS IoT SiteWise also automatically computes a set of basic aggregation metrics for all asset properties. To reduce computation costs, you can use these aggregates instead of defining custom metrics for basic computations. For more information, see Querying asset property aggregates.

Topics

- Defining metrics (console)
- Defining metrics (AWS CLI)

Defining metrics (console)

When you define a metric for an asset model in the AWS IoT SiteWise console, you specify the following parameters:

- **Name** – The property's name.
- **Data type** – The data type of the transform, which can be **Double** or **String**.
- **External ID** – (Optional) This is a user-defined ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.
- **Formula** – The metric expression. Metric expressions can use aggregation functions to input data from a property for all associated assets in a hierarchy. Start typing or press the down arrow key to open the auto complete feature. For more information, see Using formula expressions.

> **Important**
>
> Metrics can only be properties that are integer, double, Boolean, or string type. Booleans convert to `0` (false) and `1` (true).
> If you define any metric input variables in a metric's expression, those inputs must have the same time interval as the output metric.
> Formula expressions can only output double or string values. Nested expressions can output other data types, such as strings, but the formula as a whole must evaluate to a number or string. You can use the jp function to convert a string to a number. The Boolean value must be 1 (true) or 0 (false). For more information, see Undefined, infinite, and overflow values.

- **Time interval** – The metric time interval. AWS IoT SiteWise supports the following tumbling window time intervals, where each interval starts when the previous one ends:
  - **1 minute** – 1 minute, computed at the end of each minute (12:00:00 AM, 12:01:00 AM, 12:02:00 AM, and so on).
  - **5 minutes** – 5 minutes, computed at the end of every five minutes starting on the hour (12:00:00 AM, 12:05:00 AM, 12:10:00 AM, and so on).
  - **15 minutes** – 15 minutes, computed at the end of every fifteen minutes starting on the hour (12:00:00 AM, 12:15:00 AM, 12:30:00 AM, and so on).
  - **1 hour** – 1 hour (60 minutes), computed at the end of every hour in UTC (12:00:00 AM, 01:00:00 AM, 02:00:00 AM, and so on).
  - **1 day** – 1 day (24 hours), computed at the end of every day in UTC (12:00:00 AM Monday, 12:00:00 AM Tuesday, and so on).
  - **1 week** – 1 week (7 days), computed at the end of every Sunday in UTC (every 12:00:00 AM Monday).
  - **Custom interval** – You can enter any time interval between a minute and a week.
- **Offset date** – (Optional) The reference date from which to aggregate data.
- **Offset time** – (Optional) The reference time from which to aggregate data. The offset time must be between 00:00:00 and 23:59:59.
- **Offset time zone** – (Optional) The time zone for the offset. If it isn't specified, the default offset time zone is the Universal Coordinated Time (UTC).

Supported time zones

- (UTC+00:00) Universal Coordinated Time
- (UTC+01:00) European Central Time
- (UTC+02:00) Eastern European
- (UTC03+:00) Eastern African Time
- (UTC+04:00) Near East Time
- (UTC+05:00) Pakistan Lahore Time
- (UTC+05:30) India Standard Time
- (UTC+06:00) Bangladesh Standard Time
- (UTC+07:00) Vietnam Standard Time
- (UTC+08:00) China Taiwan Time
- (UTC+09:00) Japan Standard Time
- (UTC+09:30) Australia Central Time
- (UTC+10:00) Australia Eastern Time
- (UTC+11:00) Solomon Standard Time
- (UTC+12:00) New Zealand Standard Time
- (UTC-11:00) Midway Islands Time
- (UTC-10:00) Hawaii Standard Time
- (UTC-09:00) Alaska Standard Time
- (UTC-08:00) Pacific Standard Time
- (UTC-07:00) Phoenix Standard Time
- (UTC-06:00) Central Standard Time
- (UTC-05:00) Eastern Standard Time
- (UTC-04:00) Puerto Rico and US Virgin Islands Time
- (UTC-03:00) Argentina Standard Time
- (UTC-02:00) South Georgia Time
- (UTC-01:00) Central African Time

Example custom time interval with an offset (console)

The following example shows you how to define a 12-hour time interval with an offset on February 20, 2021, at 6:30:30 PM (PST).

To define a custom interval with an offset

1. For **Time interval**, choose **Custom interval**.
2. For **Time interval**, do one of the following:
   - Enter **12**, and then choose **hours**.
   - Enter **720**, and then choose **minutes**.
   - Enter **43200**, and then choose **seconds**.

   > **Important**
   >
   > The **Time interval** must be an integer regardless of the unit.
3. For **Offset date**, choose **2021/02/20**.
4. For **Offset time**, enter **18:30:30**.
5. For **Offset timezone**, choose **(UTC-08:00) Pacific Standard Time**.

If you create the metric on July 1, 2021, before or at 06:30:30 PM (PST), you get the first aggregation result on July 1, 2021, at 06:30:30 PM (PST). The second aggregation result is on July 2, 2021, at 06:30:30 AM (PST), and so on.

Defining metrics (AWS CLI)

When you define a metric for an asset model with the AWS IoT SiteWise API, you specify the following parameters:

- `name` – The property's name.
- `dataType` – The data type of the metric, which can be `DOUBLE` or `STRING`.
- `externalId` – (Optional) This is a user-defined ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

- `expression` – The metric expression. Metric expressions can use aggregation functions to input data from a property for all associated assets in a hierarchy. For more information, see Using formula expressions.
- `window` – The time interval and offset for the metric's tumbling window, where each interval starts when the previous one ends:
  - `interval` – The time interval for the tumbling window. The time interval must be between a minute and a week.
  - `offsets` – The offset for the tumbling window.

  For more information, see TumblingWindow in the *AWS IoT SiteWise API Reference*.

  Example custom time interval with an offset (AWS CLI)

  The following example shows you how to define a 12-hour time interval with an offset on February 20, 2021, at 06:30:30 PM (PST).

```
{
    "window": {
        "tumbling": {
            "interval": "12h",
            "offset": " 2021-07-23T18:30:30-08"
        }
    }
}
```

  If you create the metric on July 1, 2021, before or at 06:30:30 PM (PST), you get the first aggregation result on July 1, 2021, at 06:30:30 PM (PST). The second aggregation result is on July 2, 2021, at 06:30:30 AM (PST), and so on.
- `variables` – The list of variables that defines the other properties of your asset or child assets to use in the expression. Each variable structure contains a simple name for use in the expression and a `value` structure that identifies which property to link to that variable. The `value` structure contains the following information:
  - `propertyId` – The ID of the property from which to pull values. You can use the property's name instead of its ID if the property is defined in the current model (rather than defined in a model from a hierarchy).

- `hierarchyId` – (Optional) The ID of the hierarchy from which to query child assets for the property. You can use the hierarchy definition's name instead of its ID. If you omit this value, AWS IoT SiteWise finds the property in the current model.

> **Important**
>
> Metrics can only be properties that are integer, double, Boolean, or string type. Booleans convert to `0` (false) and `1` (true).
> If you define any metric input variables in a metric's expression, those inputs must have the same time interval as the output metric.
> Formula expressions can only output double or string values. Nested expressions can output other data types, such as strings, but the formula as a whole must evaluate to a number or string. You can use the jp function to convert a string to a number. The Boolean value must be 1 (true) or 0 (false). For more information, see Undefined, infinite, and overflow values.

- `unit` – (Optional) The scientific unit for the property, such as mm or Celsius.

Example Example metric definition

The following example demonstrates a metric property that aggregates an asset's temperature measurement data to calculate maximum hourly temperature in Fahrenheit. This object is an example of an AssetModelProperty that contains a Metric. You can specify this object as a part of the CreateAssetModel request payload to create a metric property. For more information, see Creating an asset model (AWS CLI).

```
{
      ...
      "assetModelProperties": [
      ...
      {
        "name": "Max temperature",
        "dataType": "DOUBLE",
        "type": {
          "metric": {
            "expression": "max(temp_f)",
            "variables": [
              {
```

```
                "name": "temp_f",
                "value": {
                  "propertyId": "Temperature F"
                }
              }
            ],
            "window": {
              "tumbling": {
                "interval": "1h"
              }
            }
          }
        },
        "unit": "Fahrenheit"
      }
    ],
    ...
}
```

Example Example metric definition that inputs data from associated assets

The following example demonstrates a metric property that aggregates multiple wind turbines' average power data to calculate total average power for a wind farm. This object is an example of an AssetModelProperty that contains a Metric. You can specify this object as a part of the CreateAssetModel request payload to create a metric property.

```
{
      ...
      "assetModelProperties": [
      ...
      {
          "name": "Total Average Power",
          "dataType": "DOUBLE",
          "type": {
            "metric": {
              "expression": "avg(power)",
              "variables": [
                {
                  "name": "power",
                  "value": {
                    "propertyId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
                    "hierarchyId": "Turbine Asset Model"
                  }
```

```
                }
              ],
              "window": {
                "tumbling": {
                  "interval": "5m"
                }
              }
            }
        },
        "unit": "kWh"
      }
    ],
    ...
}
```

Using formula expressions

With formula expressions, you can define the mathematical functions to transform and aggregate your raw industrial data to gain insights about your operation. Formula expressions combine literals, operators, functions, and variables to process data. For more information about how to define asset properties that use formula expressions, see Transforming data (transforms) and Aggregating data from properties and other assets (metrics). Transforms and metrics are formula properties.

Topics

- Using variables in formula expressions
- Using Literals in formula expressions
- Using operators in formula expressions
- Using constants in formula expressions
- Using functions in formula expressions
- Formula expression tutorials

Using variables in formula expressions

Variables represent AWS IoT SiteWise asset properties in formula expressions. Use variables to input values from other asset properties in your expressions, so that you can process data from constant properties (attributes), raw data streams (measurements), and other formula properties.

Variables can represent asset properties from the same asset model or from associated child asset models. Only metric formulas can input variables from child asset models.

You identify variables by different names in the console and the API.

- **AWS IoT SiteWise console** – Use asset property names as variables in your expressions.
- **AWS IoT SiteWise API (AWS CLI, AWS SDKs)** – Define variables with the ExpressionVariable structure, which requires a variable name and a reference to an asset property. The variable name can contain lowercase letters, numbers, and underscores. Then, use variable names to reference asset properties in your expressions.

Variable names are case sensitive.

For more information, see Defining transforms and Defining metrics.

Using variables to reference properties

A variable's *value* defines the property that it references. AWS IoT SiteWise provides different ways to do this.

- **By property ID:** You can specify the property's unique ID (UUID) to identify it.
- **By name:** If the property is on the same asset model, you can specify its name in the property ID field.
- **By path:** A variable value can refer to a property by its *path.* For more information, see Using paths to reference custom composite model properties.

> **Note**
>
> Variables are not supported by AWS IoT SiteWise console. They are used by AWS IoT SiteWise API, including the AWS Command Line Interface AWS CLI) and AWS SDKs.

A variable that you receive in a response from AWS IoT SiteWise includes full information about the value, including both the ID and the path.

However, when you pass a variable into AWS IoT SiteWise (for example, in a "create" or "update" call), you only need to specify one of these. For example, if you specify the path, you don't need to provide the ID.

Using Literals in formula expressions

You can define number and string literals in formula expressions.

- **Numbers**

  Use numbers and scientific notation to define integers and doubles. You can use E notation to express numbers with scientific notation.

  Examples: `1`, `2.0`, `.9`, `-23.1`, `7.89e3`, `3.4E-5`

- **Strings**

  Use the `'` (quote) and `"` (double quote) characters to define strings. The quote type for the start and end must match. To escape a quote that matches the one that you use to declare a string, include that quote character twice. This is the only escape character in AWS IoT SiteWise strings.

  Examples: `'active'`, `"inactive"`, `'{"temp": 52}'`, `"{""temp"": ""high""}"`

Using operators in formula expressions

You can use the following common operators in formula expressions.

Operator	Description
+	If both operands are numbers, this operator adds the left and right operands. If either operand is a string, this operator concatenates the left and right operands as strings. For example, the expression `1 + 2 + " is three"` evaluates to `"3 is three"`. The concatenated string can have up to 1024 characters. If the string exceeds 1024 characters, then AWS IoT SiteWise doesn't output a data point for that computation.
-	Subtracts the right operand from the left operand.

Operator	Description
	You can only use this operator with numeric operands.
`/`	Divides the left operand by the right operand. You can only use this operator with numeric operands.
`*`	Multiplies the left and right operands. You can only use this operator with numeric operands.
`^`	Raises the left operand to the power of the right operand (exponentiation). You can only use this operator with numeric operands.
`%`	Returns the remainder from dividing the left operand by the right operand. The result has the same sign as the left operand. This behavior differs from the modulo operation. You can only use this operator with numeric operands.
`x < y`	Returns `1` if `x` is less than `y`, otherwise `0`.
`x > y`	Returns `1` if `x` is greater than `y`, otherwise `0`.
`x <= y`	Returns `1` if `x` is less than or equal to `y`, otherwise `0`.
`x >= y`	Returns `1` if `x` is greater than or equal to `y`, otherwise `0`.
`x == y`	Returns `1` if `x` is equal to `y`, otherwise `0`.

Operator	Description
`x != y`	Returns `1` if `x` is not equal to `y`, otherwise `0`.
`!x`	Returns `1` if `x` is evaluated to `0` (false), otherwise `0`. `x` is evaluated to false if: • `x` is a numeric operand and it's evaluated to `0`. • `x` is evaluated to an empty string. • `x` is evaluated to an empty array. • `x` is evaluated to None.
`x and y`	Returns `0` if `x` is evaluated to `0` (false). Otherwise, returns the evaluated result of `y`. `x` or `y` is evaluated to false if: • `x` or `y` is a numeric operand and it's evaluated to `0`. • `x` or `y` is evaluated to an empty string. • `x` or `y` is evaluated to an empty array. • `x` or `y` is evaluated to None.
`x or y`	Returns `1` if `x` is evaluated to `1` (true). Otherwise, returns the evaluated result of `y`. `x` or `y` is evaluated to false if: • `x` or `y` is a numeric operand and it's evaluated to `0`. • `x` or `y` is evaluated to an empty string. • `x` or `y` is evaluated to an empty array. • `x` or `y` is evaluated to None.

Operator	Description
`not x`	Returns `1` if `x` is evaluated to `0` (false), otherwise `0`. `x` is evaluated to false if: • `x` is a numeric operand and it's evaluated to `0`. • `x` is evaluated to an empty string. • `x` is evaluated to an empty array. • `x` is evaluated to None.

`[]`
 	
`s[index]`
	Returns the character at an index `index` of the string `s`. This is equivalent to the index syntax in Python.
	
	Example Examples
	
	• `"Hello!"[1]` returns `e`.
	• `"Hello!"[-2]` returns `o`.

Operator	Description

`[]`
 	
`s[start:end:step]`
	Returns a slice of the string `s`. This is equivalent to the slice syntax in Python. This operator has the following arguments:
	
	• `start` – (Optional) The inclusive start index of the slice. Defaults to `0`.
	• `end` – (Optional) The exclusive end index of the slice. Defaults to the length of the string.
	• `step` – (Optional) The number to increment for each step in the slice. For example, you can specify 2 to return a slice with every other character, or specify `-1` to reverse the slice. Defaults to 1.
	
	You can omit the `step` argument to use its default value. For example, `s[1:4:1]` is equivalent to `s[1:4]`.
	
	The arguments must be integers or the none constant. If you specify none, AWS IoT SiteWise uses the default value for that argument.
	
	Example Examples
	
	• `"Hello!"[1:4]` returns `"ell"`.
	• `"Hello!"[:2]` returns `"He"`.
	• `"Hello!"[3:]` returns `"lo!"`.
	• `"Hello!"[:-4]` returns `"He"`.
	• `"Hello!"[::2]` returns `"Hlo"`.
	• `"Hello!"[::-1]` returns `"!olleH"`.

Using constants in formula expressions

You can use the following common mathematical constants in your expressions. All constants are case insensitive.

> **Note**
>
> If you define a variable with the same name as a constant, the variable overrides the constant.

Constant	Description
`pi`	The number pi (π): `3.141592653589793`
`e`	The number e: `2.718281828459045`
`true`	Equivalent to the number 1. In AWS IoT SiteWise, Booleans convert to their number equivalents.
`false`	Equivalent to the number 0. In AWS IoT SiteWise, Booleans convert to their number equivalents.
none	Equivalent to no value. You can use this constant to output nothing as the result of a conditional expression.

Using functions in formula expressions

You can use the following functions to operate on data in your formula expressions.

Transforms and metrics support different functions. The following table indicates which types of functions are compatible with each type of formula property.

Note

You can include a maximum of 10 functions in a formula expression.

Function type	Transforms	Metrics
Using common functions in formula expressions	 Yes	 Yes
Using comparison functions in formula expressions	 Yes	 Yes
Using conditional functions in formula expressions	 Yes	 Yes
Using string functions in formula expressions	 Yes	 Yes
Using aggregation functions in formula expressions	 No	 Yes
Using temporal functions in formula expressions	 Yes	 Yes

Function type	Transforms	Metrics
Using date and time functions in formula expressions	 Yes	 Yes

Function syntax

You can use the following syntax to create functions:

Regular syntax

With the regular syntax, the function name is followed by parentheses with zero or more arguments.

`function_name``(`*`argument1, argument2, argument3, ...`*`)`. For example, functions with the regular syntax might look like `log(x)` and `contains(s, substring)`.

Uniform function call syntax (UFCS)

UFCS enables you to call functions using the syntax for method calls in object-oriented programming. With UFCS, the first argument is followed by dot (`.`), then the function name and the remaining arguments (if any) inside parentheses.

`argument1``.`*`function_name`*`(`*`argument2, argument3, ...`*`)`. For example, functions with UFCS might look like `x.log()` and `s.contains(substring)`.

You can also use UFCS to chain subsequent functions. AWS IoT SiteWise uses the evaluation result of the current function as the first argument for the next function.

For example, you can use `message.jp('$.status').lower().contains('fail')` instead of `contains(lower(jp(message, '$.status')),'fail')`.

For more information, visit the D Programming Language website.

Note

You can use UFCS for all AWS IoT SiteWise functions.

AWS IoT SiteWise functions are not case sensitive. For example, you can use `lower(s)` and `Lower(s)` interchangeably.

Using common functions in formula expressions

In transforms and metrics, you can use the following functions to calculate common mathematical functions in transforms and metrics.

Function	Description
`abs(x)`	Returns the absolute value of `x`.
`acos(x)`	Returns the arccosine of `x`.
`asin(x)`	Returns the arcsine of `x`.
`atan(x)`	Returns the arctangent of `x`.
`cbrt(x)`	Returns the cubic root of `x`.
`ceil(x)`	Returns the nearest integer greater than `x`.
`cos(x)`	Returns the cosine of `x`.
`cosh(x)`	Returns the hyperbolic cosine of `x`.
`cot(x)`	Returns the cotangent of `x`.
`exp(x)`	Returns e to the power of `x`.
`expm1(x)`	Returns `exp(x) - 1`. Use this function to more accurately calculate `exp(x) - 1` for small values of `x`.
`floor(x)`	Returns the nearest integer less than `x`.
`log(x)`	Returns the $\log_e$ (base e) of `x`.
`log10(x)`	Returns the $\log_{10}$ (base 10) of `x`.

Function	Description
`log1p(x)`	Returns `log(1 + x)`. Use this function to more accurately calculate `log(1 + x)` for small values of `x`.
`log2(x)`	Returns the $\log_2$ (base 2) of `x`.
`pow(x, y)`	Returns `x` to the power of `y`. This is equivalent to `x ^ y`.
`signum(x)`	Returns the sign of `x` (`-1` for negative inputs, `0` for zero inputs, `+1` for positive inputs).
`sin(x)`	Returns the sine of `x`.
`sinh(x)`	Returns the hyperbolic sine of `x`.
`sqrt(x)`	Returns the square root of `x`.
`tan(x)`	Returns the tangent of `x`.
`tanh(x)`	Returns the hyperbolic tangent of `x`.

Using comparison functions in formula expressions

In transforms and metrics, you can use the following comparison functions to compare two values and output `1` (true) or `0` (false). AWS IoT SiteWise compares strings by lexicographic order.

Function	Description
`gt(x, y)`	Returns `1` if `x` is greater than `y`, otherwise `0` (`x > y`). This function doesn't return a value if `x` and `y` are incompatible types, such as a number and a string.

Function	Description
`gte(x, y)`	Returns `1` if `x` is greater than or equal to `y`, otherwise `0` (`x ≥ y`). AWS IoT SiteWise considers the arguments equal if they are within a relative tolerance of `1E-9`. This behaves similar to the isclose function in Python. This function doesn't return a value if `x` and `y` are incompatible types, such as a number and a string.
`eq(x, y)`	Returns `1` if `x` is equal to `y`, otherwise `0` (`x == y`). AWS IoT SiteWise considers the arguments equal if they are within a relative tolerance of `1E-9`. This behaves similar to the isclose function in Python. This function doesn't return a value if `x` and `y` are incompatible types, such as a number and a string.
`lt(x, y)`	Returns `1` if `x` is less than `y`, otherwise `0` (`x < y`). This function doesn't return a value if `x` and `y` are incompatible types, such as a number and a string.

Function	Description
`lte(x, y)`	Returns `1` if `x` is less than or equal to `y`, otherwise `0` (`x ≤ y`). AWS IoT SiteWise considers the arguments equal if they are within a relative tolerance of `1E-9`. This behaves similar to the isclose function in Python. This function doesn't return a value if `x` and `y` are incompatible types, such as a number and a string.
`isnan(x)`	Returns `1` if `x` is equal to `NaN`, otherwise `0`. This function doesn't return a value if `x` is a string.

Using conditional functions in formula expressions

In transforms and metrics, you can use the following function to check a condition and return different results, whether the condition evaluates to true or false.

Function	Description
`if(condition, result_if_true, result_if_false)`	Evaluates the `condition` and returns `result_if_true` if the condition evaluates to true or `result_if_false` if the condition evaluates to `false`. `condition` must be a number. This function considers `0` and an empty string as `false` and everything else (including `NaN`) as `true`. Booleans convert to `0` (false) and `1` (true). You can return the none constant from this function to discard the output for a particula

Function	Description
	r condition. This means you can filter out data points that don't meet a condition. For more information, see Filtering data points. **Example Examples** • `if(0, x, y)` returns the variable `y`. • `if(5, x, y)` returns the variable `x`. • `if(gt(temp, 300), x, y)` returns the variable `x` if the variable `temp` is greater than `300`. • `if(gt(temp, 300), temp, none)` returns the variable `temp` if it's greater than or equal to `300`, or `none` (no value) if `temp` is less than `300`. We recommend that you use UFCS for nested conditional functions where one or more arguments are conditional functions. You can use `if(condition, result_if_true)` to evaluate a condition and `elif(condition, result_if_true, result_if_false)` to evaluate additional conditions. For example, you can use `if(condition1, result1_if_true).elif(condition2, result2_if_true, result2_if_false)` instead of `if(condition1, result1_if_true, if(condition2, result2_if_true, result2_if_false))` . You can also chain additional intermediate conditional functions. For example, you can use `if(condition1, result1_i`

Function	Description
	`f_true).elif(condition2, result2_if_true).elif(condition3, result3_if_true, result3_if_false)` instead of nesting multiple `if` statements, such as `if(condition1, result1_if_true, if(condition2, result2_if_true, if(condition3, result3_if_true result3_if_false)))` . **Important** You must use `elif(condition, result_if_true, result_if_false)` with UFCS.

Using string functions in formula expressions

In transforms and metrics, you can use the following functions to operate on strings. For more information, see Using strings in formulas.

Important

Formula expressions can only output double or string values. Nested expressions can output other data types, such as strings, but the formula as a whole must evaluate to a number or string. You can use the jp function to convert a string to a number. The Boolean value must be 1 (true) or 0 (false). For more information, see Undefined, infinite, and overflow values.

Function	Description
`len(s)`	Returns the length of the string `s`.

Function	Description
`find(s, substring)`	Returns the index of the string `substring` in the string `s`.
`contains(s, substring)`	Returns `1` if the string `s` contains the string `substring`, otherwise `0`.
`upper(s)`	Returns the string `s` in uppercase form.
`lower(s)`	Returns the string `s` in lowercase form.

Function	Description
`jp(s, json_path)`	Evaluates the string `s` with the JsonPath expression `json_path` and returns the result. Use this function to do the following: • Extract a value, array, or object from a serialized JSON structure. • Convert a string to a number. For example, the formula `jp('111', '$')` returns 111 as a number. To extract a string value from a JSON structure and return it as a number, you must use multiple nested `jp` functions. The outer `jp` function extracts the string from the JSON structure, and the inner `jp` function converts the string to a number. The string `json_path` must contain a string literal. This means that `json_path` can't be an expression that evaluates to a string. **Example Examples** • `jp('{"status":"active","value":15}', '$.value')` returns 15. • `jp('{"measurement":{"reading":25,"confidence":0.95}}', '$.measurement.reading')` returns 25. • `jp('[2,8,23]', '$[2]')` returns 23. • `jp('{"values":[3,6,7]}', '$.values[1]')` returns 6. • `jp('111', '$')` returns 111.

Function	Description
	• `jp(jp('{"measurement":{"reading":25,"confidence":"0.95"}}', '$.measurement.confidence'), '$')` returns `0.95`.
`join(s0, s1, s2, s3, ...)`	Returns a concatenated string with a delimiter. This function uses the first input string as a delimiter and joins the remaining input strings together. This behaves similar to the join(CharSequence delimiter, CharSequence... elements) function in Java. **Example Examples** • `join("-", "aa", "bb", "cc")` returns `aa-bb-cc`
`format(expression: "format")` or `format("format", expression)`	Returns a string in the specified format. This function evaluates `expression` to a value, and then returns the value in the specified format. This behaves similar to the format(String format, Object... args) function in Java. For more information about supported formats, see Conversions under Class Formatter in the *Java Platform, Standard Edition 7 API Specification*. **Example Examples** • `format(100+1: "d")` returns a string, `101`. • `format("The result is %d", 100+1)` returns a string, `The result is 101`.

Function	Description
`f'expression'`	Returns a concatenated string. With this formatted function, you can use a simple expression to concatenate and format strings. These functions may contain nested expressio ns. You can use {} (curly braces) to interpola te expressions. This behaves similar to the formatted string literals in Python. **Example Examples** • `f'abc{1+2: "f"}d'` returns `abc3.0000 00d` . To evaluate this example expression, do the following: 1. `format(1+2: "f")` returns a floating point number, `3.000000`. 2. `join('', "abc", 1+2, 'd')` returns a string, `abc3.000000d` . You can also write the expression in the following way: `join('', "abc", format(1+2: "f"), 'd')` .

Using aggregation functions in formula expressions

In metrics only, you can use the following functions that aggregate input values over each time interval and calculate a single output value. Aggregation functions can aggregate data from associated assets.

Aggregation function arguments can be variables, number literals, temporal functions, nested expressions, or aggregation functions. The formula `max(latest(x), latest(y), latest(z))` uses an aggregation function as an argument and returns the largest current value of the `x`, `y`, and `z` properties.

You can use nested expressions in aggregation functions. When you use nested expressions, the following rules apply:

- Each argument can have only one variable.

  Example

  For example, `avg(x*(x-1))` and `sum(x/2 )/avg(y^2 )` are supported.

  For example, `min(x/y)` isn't supported.
- Each argument can have multilevel nested expressions.

  Example

  For example, `sum(avg(x^2 )/2)` is supported.
- Different arguments can have different variables.

  Example

  For example, `sum(x/2, y*2)` is supported.

> **Note**
>
> - If your expressions contain measurements, AWS IoT SiteWise uses the last values over the current time interval for the measurements to compute aggregates.
> - If your expressions contain attributes, AWS IoT SiteWise uses the latest values for the attributes to compute aggregates.

Function	Description
`avg(`x_0`, ..., `x_n`)`	Returns the mean of the given variables' values over the current time interval. This function outputs a data point only if the given variables have at least one data point over the current time interval.
`sum(`x_0`, ..., `x_n`)`	Returns the sum of the given variables' values over the current time interval.

Function	Description
	This function outputs a data point only if the given variables have at least one data point over the current time interval.
`min(x0, ..., xn)`	Returns the minimum of the given variables' values over the current time interval. This function outputs a data point only if the given variables have at least one data point over the current time interval.
`max(x0, ..., xn)`	Returns the maximum of the given variables' values over the current time interval. This function outputs a data point only if the given variables have at least one data point over the current time interval.
`count(x0, ..., xn)`	Returns the total number of data points for the given variables over the current time interval. For more information about how to count the number of data points that meet a condition, see Counting data points that match a condition. This function computes a data point for every time interval.
`stdev(x0, ..., xn)`	Returns the standard deviation of the given variables' values over the current time interval. This function outputs a data point only if the given variables have at least one data point over the current time interval.

Using temporal functions in formula expressions

Use temporal functions to return values based on timestamps of data points.

Using temporal functions in metrics

In metrics only, you can use the following functions that return values based on timestamps of data points.

Temporal function arguments must be properties from the local asset model or nested expressions. This means that you can't use properties from child asset models in temporal functions.

You can use nested expressions in temporal functions. When you use nested expressions, the following rules apply:

- Each argument can have only one variable.

  For example, `latest( t*9/5 + 32 )` is supported.
- Arguments can't be aggregation functions.

  For example, `first( sum(x) )` isn't supported.

Function	Description
`first(x)`	Returns the given variable's value with the earliest timestamp over the current time interval.
`last(x)`	Returns the given variable's value with the latest timestamp over the current time interval.
`earliest(x)`	Returns the given variable's last value before the start of the current time interval. This function computes a data point for every time interval, if the input property has at least one data point in its history. See time-range-defintion for details.

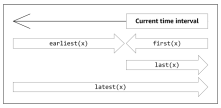
Function	Description
`latest(x)`	Returns the given variable's last value with the latest timestamp before the end of the current time interval. This function computes a data point for every time interval, if the input property has at least one data point in its history. See time-range-defintion for details.
`statetime(x)`	Returns the amount of time in seconds that the given variables are positive over the current time interval. You can use the comparison functions to create a transform property for the `statetime` function to consume. For example, if you have an `Idle` property that is `0` or `1`, you can calculate idle time per time interval with this expression: `IdleTime = statetime(Idle)` . For more informati on, see the example statetime scenario. This function doesn't support metric propertie s as input variables. This function computes a data point for every time interval, if the input property has at least one data point in its history.

Function	Description
`TimeWeightedAvg(x, [interpolation])`	Returns the average of input data weighted with time intervals between points. See Time weighted functions parameters for computation and intervals details. The optional argument `interpolaton` must be a string constant: • `locf` – This is the default. The calculation uses the Last Observed Carry Forward computation algorithm for intervals between data points. In this approach, the data point is computed as the last observed value until the next input data point time stamp. The value after a good data point is extrapolated as its value until the next data point timestamp. • `linear` – The calculation uses the linear interpolation computation algorithm for intervals between data points. The value between two good data points is extrapolated as linear interpolation between those data point's values. The value between good and bad data points or the value after the last good data point will be extrapolated as a good data point.

Function	Description
`TimeWeightedStDev(x, [algo])`	Returns the standard deviation of input data weighted with time intervals between points. See Time weighted functions parameters for computation and intervals details. The calculation uses the Last Observed Carry Forward computation algorithm for intervals between data points. In this approach, the data point is computed as the last observed value until the next input data point time stamp. Weight is computed as time interval in seconds between data points or window boundaries. The optional argument `algo` must be a string constant: • `f` – This is the default. It returns an unbiased weighted sample variance with **Frequency** weights, where TimeWeight is computed in seconds. This algorithm is usually assumed under standard deviation and is known as Bessel's correction of standard deviation for weighted samples. • p – Returns the biased weighted sample variance, also known as **Population** variance. The following formulas are used for computation where: • S_p = population standard deviation • S_f = frequency standard deviation • X_i = incoming data

Function	Description
	• ω_i = weight that equals time interval in seconds • μ* = a weighted mean of incoming data Equation for population standard deviation: $$S_p^2 = \frac{\sum_{i=1}^{N} \omega_i (x_i - \mu^*)^2}{\sum_{i=1}^{N} \omega_i}$$ Equation for frequency standard deviation: $$S_f^2 = \frac{\sum_{i=1}^{N} \omega_i (x_i - \mu^*)^2}{\sum_{i=1}^{N} \omega_i - 1}$$

The following diagram shows how AWS IoT SiteWise computes the temporal functions `first`, `last`, `earliest`, and `latest`, relative to the current time interval.



Note

- The time range for `first(x)`, `last(x)` is (current window start, current window end].
- The time range for `latest(x)` is (beginning of time, current window end].
- The time range for `earliest(x)` is (beginning of time, previous window end].

Time-weighted functions parameters

Time-weighted functions computed for the aggregate window take into account the following:

- Data points inside the window

- Time intervals between data points
- Last data point before the window
- First data point after the window (for some algorithms)

Terms:

- **Bad data point** – Any data point with non-good quality or non-number value. This is not considered in a window result computation.
- **Bad interval** – The interval after a bad data point. The interval before the first known data point is also considered a bad interval.
- **Good data point** – Any data point with good quality and numeric value.

> **Note**
>
> - AWS IoT SiteWise only consumes `GOOD` quality data when it computes transforms and metrics. It ignores `UNCERTAIN` and `BAD` data points.
> - The interval before the first known data point is considered a **bad interval**. See the section called "Formula expression tutorials" for more information.

The interval after the last known data point continues indefinitely, affecting all following windows. When a new data point arrives, the function recomputes the interval.

Following the rules above, the aggregate window result is computed and limited to window boundaries. By default, the function only sends the window result if the whole window is a **good interval**.

If the window **good interval** is smaller than the window length, the function does not send the window.

When the data points affecting the window result change, the function recalculates the window, even if the data points are outside of the window.

If the input property has at least one data point in its history and a computation has been initiated, the function calculates the time-weighted aggregate functions for every time interval.

Example Example statetime scenario

Consider an example where you have an asset with the following properties:

- `Idle` – A measurement that is `0` or 1. When the value is 1, the machine is idle.
- `Idle Time` – A metric that uses the formula `statetime(Idle)` to calculate the amount of time in seconds where the machine is idle, per 1 minute interval.

The `Idle` property has the following data points.

Timestamp	2:00:00 PM	2:00:30 PM	2:01:15 PM	2:02:45 PM	2:04:00 PM
Idle	0	1	1	0	0

AWS IoT SiteWise calculates the `Idle Time` property every minute from the values of `Idle`. After this calculation completes, the `Idle Time` property has the following data points.

Timestamp	2:00:00 PM	2:01:00 PM	2:02:00 PM	2:03:00 PM	2:04:00 PM
Idle Time	N/A	30	60	45	0

AWS IoT SiteWise performs the following calculations for `Idle Time` at the end of each minute.

- At 2:00 PM (for 1:59 PM to 2:00 PM)
  - There is no data for `Idle` before 2:00 PM, so no data point is calculated.
- At 2:01 PM (for 2:00 PM to 2:01 PM)
  - At 2:00:00 PM, the machine is active (`Idle` is `0`).
  - At 2:00:30 PM, the machine is idle (`Idle` is 1).
  - `Idle` doesn't change again before the end of the interval at 2:01:00 PM, so `Idle Time` is 30 seconds.
- At 2:02 PM (for 2:01 PM to 2:02 PM)
  - At 2:01:00 PM, the machine is idle (per the last data point at 2:00:30 PM).
  - At 2:01:15 PM, the machine is still idle.

  - `Idle` doesn't change again before the end of the interval at 2:02:00 PM, so `Idle Time` is 60 seconds.
- At 2:03 PM (for 2:02 PM to 2:03 PM)
  - At 2:02:00 PM, the machine is idle (per the last data point at 2:01:15 PM).
  - At 2:02:45 PM, the machine is active.
  - `Idle` doesn't change again before the end of the interval at 2:03:00 PM, so `Idle Time` is 45 seconds.
- At 2:04 PM (for 2:03 PM to 2:04 PM)
  - At 2:03:00 PM, the machine is active (per the last data point at 2:02:45 PM).
  - `Idle` doesn't change again before the end of the interval at 2:04:00 PM, so `Idle Time` is 0 seconds.

Example Example TimeWeightedAvg and TimeWeightedStDev scenario

The following tables provide sample inputs and outputs for these one-minute window metrics: `Avg(x)`, `TimeWeightedAvg(x)`, `TimeWeightedAvg(x, "linear")`, `stDev(x)`, `timeWeightedStDev(x)`, `timeWeightedStDev(x, 'p')`.

Sample input for one-minute aggregate window:

> **Note**
>
> These data points all have `GOOD` quality.

03:00:00	4.0
03:01:00	2.0
03:01:10	8.0
03:01:50	20.0
03:02:00	14.0
03:02:05	10.0

03:02:10	3.0
03:02:30	20.0
03:03:30	0.0

Aggregate results output:

> **Note**
>
> None – Result not produced for this window.

Time	Avg(x)	TimeWeighted Avg(x)	TimeWeighted Avg(X, "linear")	stDev(X)	timeWeighted StDev(x)	timeWeighted StDev(x, 'p')
3:00:00	4	None	None	0	None	None
3:01:00	2	4	3	0	0	0
3:02:00	14	9	13	6	5.430610041581775	5.385164807134504
3:03:00	11	13	12.875	8.54400374531753	7.724054437220943	7.659416862050705
3:04:00	0	10	2.5	0	10.084389681792215	10
3:05:00	None	0	0	None	0	0

Using temporal functions in transforms

In transforms only, you can use the `pretrigger()` function to retrieve the GOOD quality value for a variable prior to the property update that initiated the current transform calculation.

Consider an example where a manufacturer uses AWS IoT SiteWise to monitor the status of a machine. The manufacturer uses the following measurements and transforms to represent the process:

- A measurement, `current_state`, that can be 0 or 1.
  - If the machine is in the cleaning state, `current_state` equals 1.
  - If the machine is in the manufacturing state, `current_state` equals 0.
- A transform, `cleaning_state_duration`, that equals `if(pretrigger(current_state) == 1, timestamp(current_state) - timestamp(pretrigger(current_state)), none)`. This transform returns how long the machine has been in the cleaning state in seconds, in the Unix epoch format. For more information, see Using conditional functions in formula expressions and the timestamp() function.

If the machine stays in the cleaning state longer than expected, the manufacturer might investigate the machine.

You can also use the `pretrigger()` function in multivariate transforms. For example, you have two measurements named x and y, and a transform, z, that equals `x + y + pretrigger(y)`. The following table shows the values for x, y, and z from 9:00 AM to 9:15 AM.

Note

- This example assumes that the values for the measurements arrive chronologically. For example, the value of x for 09:00 AM arrives before the value of x for 09:05 AM.
- If the data points for 9:05 AM arrive before the data points for 9:00 AM, z isn't calculated at 9:05 AM.
- If the value of x for 9:05 AM arrives before the value of x for 09:00 AM and the values of y arrive chronologically, z equals `22 = 20 + 1 + 1` at 9:05 AM.

	09:00 AM	09:05 AM	09:10 AM	09:15 AM
x	10	20		30
y	1	2	3	

	09:00 AM	09:05 AM	09:10 AM	09:15 AM
`z = x + y + pretrigger(y)`	y doesn't receive any data point before 09:00 AM. Therefore, z isn't calculated at 09:00 AM.	23 = 20 + 2 + 1 `pretrigger(y)` equals 1.	25 = 20 + 3 + 2 x doesn't receive a new data point. `pretrigger(y)` equals 2.	36 = 30 + 3 + 3 y doesn't receive a new data point. Therefore, `pretrigger(y)` equals 3 at 09:15 AM.

Using date and time functions in formula expressions

In transforms and metrics, you can use the date and time functions in the following ways:

- Retrieve the current timestamp of a data point in UTC or in the local time zone.
- Construct timestamps with arguments, such as `year`, `month`, and `day_of_month`.
- Extract a time period such as a year or month with the `unix_time` argument.

Function	Description
`now()`	Returns the current date and time, in seconds, in the Unix epoch format.
`timestamp()`	• In transforms, the function returns the timestamp, in seconds, of the input message in the Unix epoch format. In transforms only, you can do one of the following: • Provide a variable as an argument to the function. The `timestamp(` *variable-name* `)` function returns the timestamp, in seconds, of the latest GOOD quality value for the specified variable in the Unix epoch format.

Function	Description
	For example, if your asset has a transform property named `Temperature_F` that uses the `9/5 * Temperature_C` formula to convert each temperature data point from Celsius to Fahrenheit, you can use the `timestamp(Temperature_F)` function to get the timestamp of the latest GOOD quality value for the `Temperature_F` property. • Use the `pretrigger()` function as an argument to the function. The `timestamp(pretrigger(` *variable-name* `))` function returns the timestamp, in seconds, of the GOOD quality value for the specified variable prior to the property update that initiated the current transform calculation in the Unix epoch format. For more information, see Using temporal functions in transforms. • In metrics, the function returns the timestamp retrieved at the end of the current window, in seconds, in the Unix epoch format.

Function	Description
`mktime(time_zone, year, month, day_of_month, hour, minute, second)`	Returns the input time in seconds, in the Unix epoch format. The following requirements apply for using this function: • The time zone argument must be a quoted string (`'UTC'`). If not specified, the default time zone is UTC. The time zone argument can be the first or last argument. • The year, month, day of month, hour, minute, and second arguments must be in order. • The year, month, and date arguments are required. The following limits apply for using this function: • `year` - Valid values are between 1970 and 2250. • `month` - Valid values are between 1 and 12. • `day-of-month` - Valid values are between 1 - 31. • `hour` - Valid values are between 0 and 23. • `minute` - Valid values are between 0 and 59. • `second` - Valid values are between 0 and 60. It can be a floating point number. Examples:

Function	Description
	• `mktime(2020, 2, 29)` • `mktime('UTC+3', 2021, 12, 31, 22)` • `mktime(2022, 10, 13, 2, 55, 13.68, 'PST')`

Function	Description
`localtime(unix_time, time_zone)`	Returns the year, the day of the month, the day of the week, the day of the year, the hour, the minute, or the second in the specified time zone from the Unix time. The following requirements apply for using this function: • The time zone argument must be a quoted string (`'UTC'`). If not specified, the default time zone is UTC. • The Unix time argument is the time in seconds, in the Unix epoch format. The valid range is between 1-31556889864403199. It can be a floating point number. Example response: `2007-12-03T10:15:30+01:00[Europe/Paris]` `localtime(unix_time, time_zone)` isn't a standalone function. The `year()`, `mon()`, `mday`, `wday()`, `yday()`, `hour()`, `minute()`, and `sec()` functions take `localtime(unix_time, time_zone)` as an argument. Examples: • `year(localtime('GMT', 1605898608.8113723))` • `now().localtime().year()` • `timestamp().localtime('PST').year()`

Function	Description
	• `localtime(1605289736, 'Europe/London').year()`
`year(localtime(unix_time, time_zone)`	Returns the year from `localtime(unix_time, time_zone)` .
`mon(localtime(unix_time, time_zone))`	Returns the month from `localtime(unix_time, time_zone)` .
`mday(localtime(unix_time, time_zone))`	Returns the day of the month from `localtime(unix_time, time_zone)` .
`wday(localtime(unix_time, time_zone))`	Returns the day of the week from `localtime(unix_time, time_zone)` .
`yday(localtime(unix_time, time_zone))`	Returns the day of the year from `localtime(unix_time, time_zone)` .
`hour(localtime(unix_time, time_zone))`	Returns the hour from `localtime(unix_time, time_zone)` .
`minute(localtime(unix_time, time_zone))`	Returns the minute from `localtime(unix_time, time_zone)` .
`sec(localtime(unix_time, time_zone))`	Returns the second from `localtime(unix_time, time_zone)` .

Supported time zone formats

You can specify the time zone argument in the following ways:

- Time zone offset - Specify `'Z'` for UTC or an offset (`'+2'` or `'-5'`).
- Offset IDs - Combine a time zone abbreviation and an offset. For example, `'GMT+2'` and `'UTC-01:00'`. The time zone abbreviation must contain only three letters.
- Region based IDs - For example, `'Etc/GMT+12'` and `'Pacific/Pago_Pago'`.

Supported time zone abbreviations

The date and time functions support the following three-letter time zone abbreviations:

- EST - -05:00
- HST - -10:00
- MST - -07:00
- ACT - Australia/Darwin
- AET - Australia/Sydney
- AGT - America/Argentina/Buenos_Aires
- ART - Africa/Cairo
- AST - America/Anchorage
- BET - America/Sao_Paulo
- BST - Asia/Dhaka
- CAT - Africa/Harare
- CET - Europe/Paris
- CNT - America/St_Johns
- CST - America/Chicago
- CTT - Asia/Shanghai
- EAT - Africa/Addis_Ababa
- IET - America/Indiana/Indianapolis
- IST - Asia/Kolkata
- JST - Asia/Tokyo
- MIT - Pacific/Apia
- NET - Asia/Yerevan
- NST - Pacific/Auckland
- PLT - Asia/Karachi
- PRT - America/Puerto_Rico
- PST - America/Los_Angeles
- SST - Pacific/Guadalcanal

- VST - Asia/Ho_Chi_Minh

Supported Region-based IDs

The date and time functions support the following Region-based IDs, organized by their relation to UTC+00:00:

- Etc/GMT+12 (UTC-12:00)
- Pacific/Pago_Pago (UTC-11:00)
- Pacific/Samoa (UTC-11:00)
- Pacific/Niue (UTC-11:00)
- US/Samoa (UTC-11:00)
- Etc/GMT+11 (UTC-11:00)
- Pacific/Midway (UTC-11:00)
- Pacific/Honolulu (UTC-10:00)
- Pacific/Rarotonga (UTC-10:00)
- Pacific/Tahiti (UTC-10:00)
- Pacific/Johnston (UTC-10:00)
- US/Hawaii (UTC-10:00)
- SystemV/HST10 (UTC-10:00)
- Etc/GMT+10 (UTC-10:00)
- Pacific/Marquesas (UTC-09:30)
- Etc/GMT+9 (UTC-09:00)
- Pacific/Gambier (UTC-09:00)
- America/Atka (UTC-09:00)
- SystemV/YST9 (UTC-09:00)
- America/Adak (UTC-09:00)
- US/Aleutian (UTC-09:00)
- Etc/GMT+8 (UTC-08:00)
- US/Alaska (UTC-08:00)

- America/Juneau (UTC-08:00)
- America/Metlakatla (UTC-08:00)
- America/Yakutat (UTC-08:00)
- Pacific/Pitcairn (UTC-08:00)
- America/Sitka (UTC-08:00)
- America/Anchorage (UTC-08:00)
- SystemV/PST8 (UTC-08:00)
- America/Nome (UTC-08:00)
- SystemV/YST9YDT (UTC-08:00)
- Canada/Yukon (UTC-07:00)
- US/Pacific-New (UTC-07:00)
- Etc/GMT+7 (UTC-07:00)
- US/Arizona (UTC-07:00)
- America/Dawson_Creek (UTC-07:00)
- Canada/Pacific (UTC-07:00)
- PST8PDT (UTC-07:00)
- SystemV/MST7 (UTC-07:00)
- America/Dawson (UTC-07:00)
- Mexico/BajaNorte (UTC-07:00)
- America/Tijuana (UTC-07:00)
- America/Creston (UTC-07:00)
- America/Hermosillo (UTC-07:00)
- America/Santa_Isabel (UTC-07:00)
- America/Vancouver (UTC-07:00)
- America/Ensenada (UTC-07:00)
- America/Phoenix (UTC-07:00)
- America/Whitehorse (UTC-07:00)
- America/Fort_Nelson (UTC-07:00)

- SystemV/PST8PDT (UTC-07:00)
- America/Los_Angeles (UTC-07:00)
- US/Pacific (UTC-07:00)
- America/El_Salvador (UTC-06:00)
- America/Guatemala (UTC-06:00)
- America/Belize (UTC-06:00)
- America/Managua (UTC-06:00)
- America/Tegucigalpa (UTC-06:00)
- Etc/GMT+6 (UTC-06:00)
- Pacific/Easter (UTC-06:00)
- Mexico/BajaSur (UTC-06:00)
- America/Regina (UTC-06:00)
- America/Denver (UTC-06:00)
- Pacific/Galapagos (UTC-06:00)
- America/Yellowknife (UTC-06:00)
- America/Swift_Current (UTC-06:00)
- America/Inuvik (UTC-06:00)
- America/Mazatlan (UTC-06:00)
- America/Boise (UTC-06:00)
- America/Costa_Rica (UTC-06:00)
- MST7MDT (UTC-06:00)
- SystemV/CST6 (UTC-06:00)
- America/Chihuahua (UTC-06:00)
- America/Ojinaga (UTC-06:00)
- Chile/EasterIsland (UTC-06:00)
- US/Mountain (UTC-06:00)
- America/Edmonton (UTC-06:00)
- Canada/Mountain (UTC-06:00)

- America/Cambridge_Bay (UTC-06:00)
- Navajo (UTC-06:00)
- SystemV/MST7MDT (UTC-06:00)
- Canada/Saskatchewan (UTC-06:00)
- America/Shiprock (UTC-06:00)
- America/Panama (UTC-05:00)
- America/Chicago (UTC-05:00)
- America/Eirunepe (UTC-05:00)
- Etc/GMT+5 (UTC-05:00)
- Mexico/General (UTC-05:00)
- America/Porto_Acre (UTC-05:00)
- America/Guayaquil (UTC-05:00)
- America/Rankin_Inlet (UTC-05:00)
- US/Central (UTC-05:00)
- America/Rainy_River (UTC-05:00)
- America/Indiana/Knox (UTC-05:00)
- America/North_Dakota/Beulah (UTC-05:00)
- America/Monterrey (UTC-05:00)
- America/Jamaica (UTC-05:00)
- America/Atikokan (UTC-05:00)
- America/Coral_Harbour (UTC-05:00)
- America/North_Dakota/Center (UTC-05:00)
- America/Cayman (UTC-05:00)
- America/Indiana/Tell_City (UTC-05:00)
- America/Mexico_City (UTC-05:00)
- America/Matamoros (UTC-05:00)
- CST6CDT (UTC-05:00)
- America/Knox_IN (UTC-05:00)

- America/Bogota (UTC-05:00)
- America/Menominee (UTC-05:00)
- America/Resolute (UTC-05:00)
- SystemV/EST5 (UTC-05:00)
- Canada/Central (UTC-05:00)
- Brazil/Acre (UTC-05:00)
- America/Cancun (UTC-05:00)
- America/Lima (UTC-05:00)
- America/Bahia_Banderas (UTC-05:00)
- US/Indiana-Starke (UTC-05:00)
- America/Rio_Branco (UTC-05:00)
- SystemV/CST6CDT (UTC-05:00)
- Jamaica (UTC-05:00)
- America/Merida (UTC-05:00)
- America/North_Dakota/New_Salem (UTC-05:00)
- America/Winnipeg (UTC-05:00)
- America/Cuiaba (UTC-04:00)
- America/Marigot (UTC-04:00)
- America/Indiana/Petersburg (UTC-04:00)
- Chile/Continental (UTC-04:00)
- America/Grand_Turk (UTC-04:00)
- Cuba (UTC-04:00)
- Etc/GMT+4 (UTC-04:00)
- America/Manaus (UTC-04:00)
- America/Fort_Wayne (UTC-04:00)
- America/St_Thomas (UTC-04:00)
- America/Anguilla (UTC-04:00)
- America/Havana (UTC-04:00)
- US/Michigan (UTC-04:00)

- America/Barbados (UTC-04:00)
- America/Louisville (UTC-04:00)
- America/Curacao (UTC-04:00)
- America/Guyana (UTC-04:00)
- America/Martinique (UTC-04:00)
- America/Puerto_Rico (UTC-04:00)
- America/Port_of_Spain (UTC-04:00)
- SystemV/AST4 (UTC-04:00)
- America/Indiana/Vevay (UTC-04:00)
- America/Indiana/Vincennes (UTC-04:00)
- America/Kralendijk (UTC-04:00)
- America/Antigua (UTC-04:00)
- America/Indianapolis (UTC-04:00)
- America/Iqaluit (UTC-04:00)
- America/St_Vincent (UTC-04:00)
- America/Kentucky/Louisville (UTC-04:00)
- America/Dominica (UTC-04:00)
- America/Asuncion (UTC-04:00)
- EST5EDT (UTC-04:00)
- America/Nassau (UTC-04:00)
- America/Kentucky/Monticello (UTC-04:00)
- Brazil/West (UTC-04:00)
- America/Aruba (UTC-04:00)
- America/Indiana/Indianapolis (UTC-04:00)
- America/Santiago (UTC-04:00)
- America/La_Paz (UTC-04:00)
- America/Thunder_Bay (UTC-04:00)
- America/Indiana/Marengo (UTC-04:00)
- America/Blanc-Sablon (UTC-04:00)

- America/Santo_Domingo (UTC-04:00)
- US/Eastern (UTC-04:00)
- Canada/Eastern (UTC-04:00)
- America/Port-au-Prince (UTC-04:00)
- America/St_Barthelemy (UTC-04:00)
- America/Nipigon (UTC-04:00)
- US/East-Indiana (UTC-04:00)
- America/St_Lucia (UTC-04:00)
- America/Montserrat (UTC-04:00)
- America/Lower_Princes (UTC-04:00)
- America/Detroit (UTC-04:00)
- America/Tortola (UTC-04:00)
- America/Porto_Velho (UTC-04:00)
- America/Campo_Grande (UTC-04:00)
- America/Virgin (UTC-04:00)
- America/Pangnirtung (UTC-04:00)
- America/Montreal (UTC-04:00)
- America/Indiana/Winamac (UTC-04:00)
- America/Boa_Vista (UTC-04:00)
- America/Grenada (UTC-04:00)
- America/New_York (UTC-04:00)
- America/St_Kitts (UTC-04:00)
- America/Caracas (UTC-04:00)
- America/Guadeloupe (UTC-04:00)
- America/Toronto (UTC-04:00)
- SystemV/EST5EDT (UTC-04:00)
- America/Argentina/Catamarca (UTC-03:00)
- Canada/Atlantic (UTC-03:00)
- America/Argentina/Cordoba (UTC-03:00)

- America/Araguaina (UTC-03:00)
- America/Argentina/Salta (UTC-03:00)
- Etc/GMT+3 (UTC-03:00)
- America/Montevideo (UTC-03:00)
- Brazil/East (UTC-03:00)
- America/Argentina/Mendoza (UTC-03:00)
- America/Argentina/Rio_Gallegos (UTC-03:00)
- America/Catamarca (UTC-03:00)
- America/Cordoba (UTC-03:00)
- America/Sao_Paulo (UTC-03:00)
- America/Argentina/Jujuy (UTC-03:00)
- America/Cayenne (UTC-03:00)
- America/Recife (UTC-03:00)
- America/Buenos_Aires (UTC-03:00)
- America/Paramaribo (UTC-03:00)
- America/Moncton (UTC-03:00)
- America/Mendoza (UTC-03:00)
- America/Santarem (UTC-03:00)
- Atlantic/Bermuda (UTC-03:00)
- America/Maceio (UTC-03:00)
- Atlantic/Stanley (UTC-03:00)
- America/Halifax (UTC-03:00)
- Antarctica/Rothera (UTC-03:00)
- America/Argentina/San_Luis (UTC-03:00)
- America/Argentina/Ushuaia (UTC-03:00)
- Antarctica/Palmer (UTC-03:00)
- America/Punta_Arenas (UTC-03:00)
- America/Glace_Bay (UTC-03:00)
- America/Fortaleza (UTC-03:00)

- America/Thule (UTC-03:00)
- America/Argentina/La_Rioja (UTC-03:00)
- America/Belem (UTC-03:00)
- America/Jujuy (UTC-03:00)
- America/Bahia (UTC-03:00)
- America/Goose_Bay (UTC-03:00)
- America/Argentina/San_Juan (UTC-03:00)
- America/Argentina/ComodRivadavia (UTC-03:00)
- America/Argentina/Tucuman (UTC-03:00)
- America/Rosario (UTC-03:00)
- SystemV/AST4ADT (UTC-03:00)
- America/Argentina/Buenos_Aires (UTC-03:00)
- America/St_Johns (UTC-02:30)
- Canada/Newfoundland (UTC-02:30)
- America/Miquelon (UTC-02:00)
- Etc/GMT+2 (UTC-02:00)
- America/Godthab (UTC-02:00)
- America/Noronha (UTC-02:00)
- Brazil/DeNoronha (UTC-02:00)
- Atlantic/South_Georgia (UTC-02:00)
- Etc/GMT+1 (UTC-01:00)
- Atlantic/Cape_Verde (UTC-01:00)
- Pacific/Kiritimati (UTC+14:00)
- Etc/GMT-14 (UTC+14:00)
- Pacific/Fakaofo (UTC+13:00)
- Pacific/Enderbury (UTC+13:00)
- Pacific/Apia (UTC+13:00)
- Pacific/Tongatapu (UTC+13:00)
- Etc/GMT-13 (UTC+13:00)

- NZ-CHAT (UTC+12:45)
- Pacific/Chatham (UTC+12:45)
- Pacific/Kwajalein (UTC+12:00)
- Antarctica/McMurdo (UTC+12:00)
- Pacific/Wallis (UTC+12:00)
- Pacific/Fiji (UTC+12:00)
- Pacific/Funafuti (UTC+12:00)
- Pacific/Nauru (UTC+12:00)
- Kwajalein (UTC+12:00)
- NZ (UTC+12:00)
- Pacific/Wake (UTC+12:00)
- Antarctica/South_Pole (UTC+12:00)
- Pacific/Tarawa (UTC+12:00)
- Pacific/Auckland (UTC+12:00)
- Asia/Kamchatka (UTC+12:00)
- Etc/GMT-12 (UTC+12:00)
- Asia/Anadyr (UTC+12:00)
- Pacific/Majuro (UTC+12:00)
- Pacific/Ponape (UTC+11:00)
- Pacific/Bougainville (UTC+11:00)
- Antarctica/Macquarie (UTC+11:00)
- Pacific/Pohnpei (UTC+11:00)
- Pacific/Efate (UTC+11:00)
- Pacific/Norfolk (UTC+11:00)
- Asia/Magadan (UTC+11:00)
- Pacific/Kosrae (UTC+11:00)
- Asia/Sakhalin (UTC+11:00)
- Pacific/Noumea (UTC+11:00)
- Etc/GMT-11 (UTC+11:00)

- Asia/Srednekolymsk (UTC+11:00)
- Pacific/Guadalcanal (UTC+11:00)
- Australia/Lord_Howe (UTC+10:30)
- Australia/LHI (UTC+10:30)
- Australia/Hobart (UTC+10:00)
- Pacific/Yap (UTC+10:00)
- Australia/Tasmania (UTC+10:00)
- Pacific/Port_Moresby (UTC+10:00)
- Australia/ACT (UTC+10:00)
- Australia/Victoria (UTC+10:00)
- Pacific/Chuuk (UTC+10:00)
- Australia/Queensland (UTC+10:00)
- Australia/Canberra (UTC+10:00)
- Australia/Currie (UTC+10:00)
- Pacific/Guam (UTC+10:00)
- Pacific/Truk (UTC+10:00)
- Australia/NSW (UTC+10:00)
- Asia/Vladivostok (UTC+10:00)
- Pacific/Saipan (UTC+10:00)
- Antarctica/DumontDUrville (UTC+10:00)
- Australia/Sydney (UTC+10:00)
- Australia/Brisbane (UTC+10:00)
- Etc/GMT-10 (UTC+10:00)
- Asia/Ust-Nera (UTC+10:00)
- Australia/Melbourne (UTC+10:00)
- Australia/Lindeman (UTC+10:00)
- Australia/North (UTC+09:30)
- Australia/Yancowinna (UTC+09:30)
- Australia/Adelaide (UTC+09:30)

- Australia/Broken_Hill (UTC+09:30)
- Australia/South (UTC+09:30)
- Australia/Darwin (UTC+09:30)
- Etc/GMT-9 (UTC+09:00)
- Pacific/Palau (UTC+09:00)
- Asia/Chita (UTC+09:00)
- Asia/Dili (UTC+09:00)
- Asia/Jayapura (UTC+09:00)
- Asia/Yakutsk (UTC+09:00)
- Asia/Pyongyang (UTC+09:00)
- ROK (UTC+09:00)
- Asia/Seoul (UTC+09:00)
- Asia/Khandyga (UTC+09:00)
- Japan (UTC+09:00)
- Asia/Tokyo (UTC+09:00)
- Australia/Eucla (UTC+08:45)
- Asia/Kuching (UTC+08:00)
- Asia/Chungking (UTC+08:00)
- Etc/GMT-8 (UTC+08:00)
- Australia/Perth (UTC+08:00)
- Asia/Macao (UTC+08:00)
- Asia/Macau (UTC+08:00)
- Asia/Choibalsan (UTC+08:00)
- Asia/Shanghai (UTC+08:00)
- Antarctica/Casey (UTC+08:00)
- Asia/Ulan_Bator (UTC+08:00)
- Asia/Chongqing (UTC+08:00)
- Asia/Ulaanbaatar (UTC+08:00)
- Asia/Taipei (UTC+08:00)

- Asia/Manila (UTC+08:00)
- PRC (UTC+08:00)
- Asia/Ujung_Pandang (UTC+08:00)
- Asia/Harbin (UTC+08:00)
- Singapore (UTC+08:00)
- Asia/Brunei (UTC+08:00)
- Australia/West (UTC+08:00)
- Asia/Hong_Kong (UTC+08:00)
- Asia/Makassar (UTC+08:00)
- Hongkong (UTC+08:00)
- Asia/Kuala_Lumpur (UTC+08:00)
- Asia/Irkutsk (UTC+08:00)
- Asia/Singapore (UTC+08:00)
- Asia/Pontianak (UTC+07:00)
- Etc/GMT-7 (UTC+07:00)
- Asia/Phnom_Penh (UTC+07:00)
- Asia/Novosibirsk (UTC+07:00)
- Antarctica/Davis (UTC+07:00)
- Asia/Tomsk (UTC+07:00)
- Asia/Jakarta (UTC+07:00)
- Asia/Barnaul (UTC+07:00)
- Indian/Christmas (UTC+07:00)
- Asia/Ho_Chi_Minh (UTC+07:00)
- Asia/Hovd (UTC+07:00)
- Asia/Bangkok (UTC+07:00)
- Asia/Vientiane (UTC+07:00)
- Asia/Novokuznetsk (UTC+07:00)
- Asia/Krasnoyarsk (UTC+07:00)
- Asia/Saigon (UTC+07:00)

- Asia/Yangon (UTC+06:30)
- Asia/Rangoon (UTC+06:30)
- Indian/Cocos (UTC+06:30)
- Asia/Kashgar (UTC+06:00)
- Etc/GMT-6 (UTC+06:00)
- Asia/Almaty (UTC+06:00)
- Asia/Dacca (UTC+06:00)
- Asia/Omsk (UTC+06:00)
- Asia/Dhaka (UTC+06:00)
- Indian/Chagos (UTC+06:00)
- Asia/Qyzylorda (UTC+06:00)
- Asia/Bishkek (UTC+06:00)
- Antarctica/Vostok (UTC+06:00)
- Asia/Urumqi (UTC+06:00)
- Asia/Thimbu (UTC+06:00)
- Asia/Thimphu (UTC+06:00)
- Asia/Kathmandu (UTC+05:45)
- Asia/Katmandu (UTC+05:45)
- Asia/Kolkata (UTC+05:30)
- Asia/Colombo (UTC+05:30)
- Asia/Calcutta (UTC+05:30)
- Asia/Aqtau (UTC+05:00)
- Etc/GMT-5 (UTC+05:00)
- Asia/Samarkand (UTC+05:00)
- Asia/Karachi (UTC+05:00)
- Asia/Yekaterinburg (UTC+05:00)
- Asia/Dushanbe (UTC+05:00)
- Indian/Maldives (UTC+05:00)
- Asia/Oral (UTC+05:00)

- Asia/Tashkent (UTC+05:00)
- Antarctica/Mawson (UTC+05:00)
- Asia/Aqtobe (UTC+05:00)
- Asia/Ashkhabad (UTC+05:00)
- Asia/Ashgabat (UTC+05:00)
- Asia/Atyrau (UTC+05:00)
- Indian/Kerguelen (UTC+05:00)
- Iran (UTC+04:30)
- Asia/Tehran (UTC+04:30)
- Asia/Kabul (UTC+04:30)
- Asia/Yerevan (UTC+04:00)
- Etc/GMT-4 (UTC+04:00)
- Etc/GMT-4 (UTC+04:00)
- Asia/Dubai (UTC+04:00)
- Indian/Reunion (UTC+04:00)
- Europe/Saratov (UTC+04:00)
- Europe/Samara (UTC+04:00)
- Indian/Mahe (UTC+04:00)
- Asia/Baku (UTC+04:00)
- Asia/Muscat (UTC+04:00)
- Europe/Volgograd (UTC+04:00)
- Europe/Astrakhan (UTC+04:00)
- Asia/Tbilisi (UTC+04:00)
- Europe/Ulyanovsk (UTC+04:00)
- Asia/Aden (UTC+03:00)
- Africa/Nairobi (UTC+03:00)
- Europe/Istanbul (UTC+03:00)
- Etc/GMT-3 (UTC+03:00)
- Europe/Zaporozhye (UTC+03:00)

- Israel (UTC+03:00)
- Indian/Comoro (UTC+03:00)
- Antarctica/Syowa (UTC+03:00)
- Africa/Mogadishu (UTC+03:00)
- Europe/Bucharest (UTC+03:00)
- Africa/Asmera (UTC+03:00)
- Europe/Mariehamn (UTC+03:00)
- Asia/Istanbul (UTC+03:00)
- Europe/Tiraspol (UTC+03:00)
- Europe/Moscow (UTC+03:00)
- Europe/Chisinau (UTC+03:00)
- Europe/Helsinki (UTC+03:00)
- Asia/Beirut (UTC+03:00)
- Asia/Tel_Aviv (UTC+03:00)
- Africa/Djibouti (UTC+03:00)
- Europe/Simferopol (UTC+03:00)
- Europe/Sofia (UTC+03:00)
- Asia/Gaza (UTC+03:00)
- Africa/Asmara (UTC+03:00)
- Europe/Riga (UTC+03:00)
- Asia/Baghdad (UTC+03:00)
- Asia/Damascus (UTC+03:00)
- Africa/Dar_es_Salaam (UTC+03:00)
- Africa/Addis_Ababa (UTC+03:00)
- Europe/Uzhgorod (UTC+03:00)
- Asia/Jerusalem (UTC+03:00)
- Asia/Riyadh (UTC+03:00)
- Asia/Kuwait (UTC+03:00)
- Europe/Kirov (UTC+03:00)

- Africa/Kampala (UTC+03:00)
- Europe/Minsk (UTC+03:00)
- Asia/Qatar (UTC+03:00)
- Europe/Kiev (UTC+03:00)
- Asia/Bahrain (UTC+03:00)
- Europe/Vilnius (UTC+03:00)
- Indian/Antananarivo (UTC+03:00)
- Indian/Mayotte (UTC+03:00)
- Europe/Tallinn (UTC+03:00)
- Turkey (UTC+03:00)
- Africa/Juba (UTC+03:00)
- Asia/Nicosia (UTC+03:00)
- Asia/Famagusta (UTC+03:00)
- W-SU (UTC+03:00)
- EET (UTC+03:00)
- Asia/Hebron (UTC+03:00)
- Asia/Amman (UTC+03:00)
- Europe/Nicosia (UTC+03:00)
- Europe/Athens (UTC+03:00)
- Africa/Cairo (UTC+02:00)
- Africa/Mbabane (UTC+02:00)
- Europe/Brussels (UTC+02:00)
- Europe/Warsaw (UTC+02:00)
- CET (UTC+02:00)
- Europe/Luxembourg (UTC+02:00)
- Etc/GMT-2 (UTC+02:00)
- Libya (UTC+02:00)
- Africa/Kigali (UTC+02:00)
- Africa/Tripoli (UTC+02:00)

- Europe/Kaliningrad (UTC+02:00)
- Africa/Windhoek (UTC+02:00)
- Europe/Malta (UTC+02:00)
- Europe/Busingen (UTC+02:00)
-
- Europe/Skopje (UTC+02:00)
- Europe/Sarajevo (UTC+02:00)
- Europe/Rome (UTC+02:00)
- Europe/Zurich (UTC+02:00)
- Europe/Gibraltar (UTC+02:00)
- Africa/Lubumbashi (UTC+02:00)
- Europe/Vaduz (UTC+02:00)
- Europe/Ljubljana (UTC+02:00)
- Europe/Berlin (UTC+02:00)
- Europe/Stockholm (UTC+02:00)
- Europe/Budapest (UTC+02:00)
- Europe/Zagreb (UTC+02:00)
- Europe/Paris (UTC+02:00)
- Africa/Ceuta (UTC+02:00)
- Europe/Prague (UTC+02:00)
- Antarctica/Troll (UTC+02:00)
- Africa/Gaborone (UTC+02:00)
- Europe/Copenhagen (UTC+02:00)
- Europe/Vienna (UTC+02:00)
- Europe/Tirane (UTC+02:00)
- MET (UTC+02:00)
- Europe/Amsterdam (UTC+02:00)
- Africa/Maputo (UTC+02:00)
- Europe/San_Marino (UTC+02:00)

- Poland (UTC+02:00)
- Europe/Andorra (UTC+02:00)
- Europe/Oslo (UTC+02:00)
- Europe/Podgorica (UTC+02:00)
- Africa/Bujumbura (UTC+02:00)
- Atlantic/Jan_Mayen (UTC+02:00)
- Africa/Maseru (UTC+02:00)
- Europe/Madrid (UTC+02:00)
- Africa/Blantyre (UTC+02:00)
- Africa/Lusaka (UTC+02:00)
- Africa/Harare (UTC+02:00)
- Africa/Khartoum (UTC+02:00)
- Africa/Johannesburg (UTC+02:00)
- Europe/Belgrade (UTC+02:00)
- Europe/Bratislava (UTC+02:00)
- Arctic/Longyearbyen (UTC+02:00)
- Egypt (UTC+02:00)
- Europe/Vatican (UTC+02:00)
- Europe/Monaco (UTC+02:00)
- Europe/London (UTC+01:00)
- Etc/GMT-1 (UTC+01:00)
- Europe/Jersey (UTC+01:00)
- Europe/Guernsey (UTC+01:00)
- Europe/Isle_of_Man (UTC+01:00)
- Africa/Tunis (UTC+01:00)
- Africa/Malabo (UTC+01:00)
- GB-Eire (UTC+01:00)
- Africa/Lagos (UTC+01:00)
- Africa/Algiers (UTC+01:00)

- GB (UTC+01:00)
- Portugal (UTC+01:00)
- Africa/Sao_Tome (UTC+01:00)
- Africa/Ndjamena (UTC+01:00)
- Atlantic/Faeroe (UTC+01:00)
- Eire (UTC+01:00)
- Atlantic/Faroe (UTC+01:00)
- Europe/Dublin (UTC+01:00)
- Africa/Libreville (UTC+01:00)
- Africa/El_Aaiun (UTC+01:00)
- Africa/El_Aaiun (UTC+01:00)
- Africa/Douala (UTC+01:00)
- Africa/Brazzaville (UTC+01:00)
- Africa/Porto-Novo (UTC+01:00)
- Atlantic/Madeira (UTC+01:00)
- Europe/Lisbon (UTC+01:00)
- Atlantic/Canary (UTC+01:00)
- Africa/Casablanca (UTC+01:00)
- Europe/Belfast (UTC+01:00)
- Africa/Luanda (UTC+01:00)
- Africa/Kinshasa (UTC+01:00)
- Africa/Bangui (UTC+01:00)
- WET (UTC+01:00)
- Africa/Niamey (UTC+01:00)
- GMT (UTC+00:00)
- Etc/GMT-0 (UTC+00:00)
- Atlantic/St_Helena (UTC+00:00)
- Etc/GMT+0 (UTC+00:00)
- Africa/Banjul (UTC+00:00)

- Etc/GMT (UTC+00:00)
- Africa/Freetown (UTC+00:00)
- Africa/Bamako (UTC+00:00)
- Africa/Conakry (UTC+00:00)
- Universal (UTC+00:00)
- Africa/Nouakchott (UTC+00:00)
- UTC (UTC+00:00)
- Etc/Universal (UTC+00:00)
- Atlantic/Azores (UTC+00:00)
- Africa/Abidjan (UTC+00:00)
- Africa/Accra (UTC+00:00)
- Etc/UCT (UTC+00:00)
- GMT0 (UTC+00:00)
- Zulu (UTC+00:00)Zulu (UTC+00:00)
- Africa/Ouagadougou (UTC+00:00)
- Atlantic/Reykjavik (UTC+00:00)
- Etc/Zulu (UTC+00:00)
- Iceland (UTC+00:00)
- Africa/Lome (UTC+00:00)
- Greenwich (UTC+00:00)
- Etc/GMT0 (UTC+00:00)
- America/Danmarkshavn (UTC+00:00)
- Africa/Dakar (UTC+00:00)
- Africa/Bissau (UTC+00:00)
- Etc/Greenwich (UTC+00:00)
- Africa/Timbuktu (UTC+00:00)
- UCT (UTC+00:00)
- Africa/Monrovia (UTC+00:00)
- Etc/UTC (UTC+00:00)

Formula expression tutorials

You can follow these tutorials to use formula expressions in AWS IoT SiteWise.

Topics

- Using strings in formulas
- Filtering data points
- Counting data points that match a condition
- Late data in formulas
- Data quality in formulas
- Undefined, infinite, and overflow values

Using strings in formulas

You can operate on strings in your formula expressions. You also can input strings from variables that reference attribute and measurement properties.

> **Important**
>
> Formula expressions can only output double or string values. Nested expressions can output other data types, such as strings, but the formula as a whole must evaluate to a number or string. You can use the jp function to convert a string to a number. The Boolean value must be 1 (true) or 0 (false). For more information, see Undefined, infinite, and overflow values.

AWS IoT SiteWise provides the following formula expression features that you can use to operate on strings:

- String literals
- The index operator (`s[index]`)
- The slice operator (`s[start:end:step]`)
- Comparison functions, which you can use compare strings by lexicographic order
- String functions, which include the `jp` function that can parse serialized JSON objects and convert strings to numbers

Filtering data points

You can use the if function to filter out data points that don't meet a condition. The `if` function evaluates a condition and returns different values for `true` and `false` results. You can use the none constant as an output for one case of an `if` function to discard the data point for that case.

To filter out data points that match a condition

- Create a transform that uses the `if` function to define a condition that checks if a condition is met, and returns none as either the `result_if_true` or `result_if_false` value.

Example Example: Filter out data points where water isn't boiling

Consider a scenario where you have a measurement, `temp_c`, that provides the temperature (in Celsius) of water in a machine. You can define the following transform to filter out data points where the water isn't boiling:

- Transform: `boiling_temps = if(gte(temp_c, 100), temp_c, none)` – Returns the temperature if it's greater than or equal to 100 degrees Celsius, otherwise returns no data point.

Counting data points that match a condition

You can use comparison functions and sum() to count the number of data points for which a condition is true.

To count data points that match a condition

1. Create a transform that uses a comparison function to define a filter condition on another property.
2. Create a metric that sums the data points where that condition is met.

Example Example: Count the number of data points where water is boiling

Consider a scenario where you have a measurement, `temp_c`, that provides the temperature (in Celsius) of water in a machine. You can define the following transform and metric properties to count the number of data points where the water is boiling:

- Transform: `is_boiling = gte(temp_c, 100)` – Returns 1 if the temperature is greater than or equal to 100 degrees Celsius, otherwise returns `0`.

- Metric: `boiling_count = sum(is_boiling)` – Returns the number of data points where water is boiling.

Late data in formulas

AWS IoT SiteWise supports late data ingestion of data that is up to 7 days old. When AWS IoT SiteWise receives late data, it recalculates existing values for any metric that inputs the late data in a past window. These recalculations result in data processing charges.

> **Note**
>
> When AWS IoT SiteWise computes properties that input late data, it uses each property's current formula expression.

After AWS IoT SiteWise recalculates a past window for a metric, it replaces the previous value for that window. If you enabled notifications for that metric, AWS IoT SiteWise also emits a property value notification. This means that you can receive a new property value update notification for the same property and timestamp for which you previously received a notification. If your applications or data lakes consume property value notifications, you must update the previous value with the new value so that their data is accurate.

Data quality in formulas

In AWS IoT SiteWise, each data point has a quality code, which can be one of the following:

- `GOOD` – The data isn't affected by any issues.
- `BAD` – The data is affected by an issue such as sensor failure.
- `UNCERTAIN` – The data is affected by an issue such as sensor inaccuracy.

AWS IoT SiteWise consumes only `GOOD` quality data when it computes transforms and metrics. AWS IoT SiteWise outputs only `GOOD` quality data for successful computations. If a computation is unsuccessful, then AWS IoT SiteWise doesn't output a data point for that computation. This can occur if a computation results in an undefined, infinite, or overflow value.

For more information about how to query data and filter by data quality, see Query data from AWS IoT SiteWise.

Undefined, infinite, and overflow values

Some formula expressions (such as `x / 0`, `sqrt(-1)`, or `log(0)`) calculate values that are undefined in a real number system, infinite, or outside the range supported by AWS IoT SiteWise. When an asset property's expression computes an undefined, infinite, or overflow value, AWS IoT SiteWise doesn't output a data point for that computation.

AWS IoT SiteWise also doesn't output a data point if it computes a non-numeric value as the result of a formula expression. This means that if you define a formula that computes a string, array, or the none constant, then AWS IoT SiteWise doesn't output a data point for that computation.

Example Examples

Each of the following formula expressions result in a value that AWS IoT SiteWise can't represent as a number. AWS IoT SiteWise doesn't output a data point when it computes these formula expressions.

- `x / 0` is undefined.
- `log(0)` is undefined.
- `sqrt(-1)` is undefined in a real number system.
- `"hello" + " world"` is a string.
- `jp('{"values":[3,6,7]}', '$.values')` is an array.
- `if(gte(temp, 300), temp, none)` is none when `temp` is less than `300`.

Creating custom composite models (Components)

Custom composite models, or components if you're using the console, provide another level of organization for your asset models and component models. You can use them to structure your models by grouping properties or referencing other models. For more information about working with custom composite models, see Custom composite models (Components).

You create a custom composite model within an existing asset model or component model. There are two types of custom composite models. To group related properties within a model, you can create an **inline** custom composite model. To reference a component model within your asset model or component model, you can create a **component-model-based** custom composite model.

The following sections describe how to use the AWS IoT SiteWise API to create custom composite models.

Topics

Creating an inline component (console)

You can use the AWS IoT SiteWise console to create an inline component that defines its own properties.

> **Note**
>
> Because this is an *inline* component, these properties only apply to the current asset model and aren't shared anywhere else.
> If you need to produce a reusable model (for example, to share among multiple asset models, or to include multiple instances within one asset model), you should create a component based on a component model instead. See the following section for details.

To create a component (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Models**.
3. Choose the asset model to which you want to add a component.
4. On the **Properties** tab, choose **Components**.
5. Choose **Create component**.
6. On the **Create component** page, do the following:

    a. Enter a **Name** for the component, such as **`ServoMotor`** or **`ServoMotor Model`**. This name must be unique across all components in your account in this Region.

    b. (Optional) Add **Attribute definitions** for the model. Attributes represent information that rarely changes. For more information, see Defining static data (attributes).

c. (Optional) Add **Measurement definitions** for the model. Measurements represent data streams from your equipment. For more information, see Defining data streams from equipment (measurements).

d. (Optional) Add **Transform definitions** for the model. Transforms are formulas that map data from one form to another. For more information, see Transforming data (transforms).

e. (Optional) Add **Metric definitions** for the model. Metrics are formulas that aggregate data over time intervals. Metrics can input data from associated assets, so that you can calculate values that represent your operation or a subset of your operation. For more information, see Aggregating data from properties and other assets (metrics).

f. Choose **Create component**.

Creating an inline custom composite model (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to create an inline custom composite model that defines its own properties.

Use the CreateAssetModelCompositeModel operation to create an inline model with properties. This operation expects a payload with the following structure.

> **Note**
>
> Because this is an *inline* composite model, these properties only apply to the current asset model and aren't shared anywhere else. What makes it "inline" is that it doesn't provide a value for the `composedAssetModelId` field.
> If you need to produce a reusable model (for example, to share among multiple asset models, or to include multiple instances within one asset model), you should create a *component-model-based* composite model instead. See the following section for details.

```
{
    "assetModelCompositeModelName": "CNCLathe_ServoMotorA",
    "assetModelCompositeModelType": "CUSTOM",
    "assetModelCompositeModelProperties": [
        {
            "dataType": "DOUBLE",
            "name": "Servo Motor Temperature",
            "type": {
```

```
            "measurement": {}
            },
            "unit": "Celsius"
        },
        {
            "dataType": "DOUBLE",
            "name": "Spindle speed",
            "type": {
            "measurement": {}
            },
            "unit": "rpm"
        }
    ]
}
```

Creating a component-model-based component (console)

You can use the AWS IoT SiteWise console to create a component based on a component model.

To create a component-model-based component (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Models**.
3. Choose the asset model to which you want to add a component.
4. On the **Properties** tab, choose **Components**.
5. Choose **Create component**.
6. On the **Create component** page, do the following:
    a. Select the component model you want to based the component on.
    b. Enter a **Name** for the component, such as **`ServoMotor`** or **`ServoMotor Model`**. This name must be unique across all components in your account in this Region.
    c. Choose **Create component**.

Creating a component-model-based custom composite model (AWS CLI)

You can use the AWS CLI to create a component-model-based custom composite model within your asset model. A component-model-based custom composite model is a reference to a component model that you've already defined elsewhere.

Use the CreateAssetModelCompositeModel operation to create a component-model-based custom composite model. This operation expects a payload with the following structure.

> **Note**
>
> In this example, the value of `composedAssetModelId` is the asset model ID or external ID of an existing component model. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*. For an example of how to create a component model, see Creating a component model (AWS CLI).

```
{
    "assetModelCompositeModelName": "CNCLathe_ServoMotorA",
    "assetModelCompositeModelType": "CUSTOM",
    "composedAssetModelId": component model ID
]
```

Since it's just a reference, a component-model-based custom composite model has no properties of its own, other than a name.

If you want to add multiple instances of the same component to your asset model (for example, a CNC machine that has multiple servo motors), you can add multiple component-model-based custom composite models that each have their own name but which all reference the same `composedAssetModelId`.

You can nest components within other components. To do so, you can add a component-model-based composite model, as shown in this example, to one of your component models.

Creating assets

You can create an asset from an asset model. You must have an asset model before you can create an asset. If you haven't created an asset model, see Creating asset models.

> **Note**
>
> You can only create assets from `ACTIVE` models. If your model's state isn't `ACTIVE`, you may need to wait for up to a few minutes before you can create assets from that model. For more information, see Asset and model states.

Topics

Creating an asset (console)

You can use the AWS IoT SiteWise console to create an asset.

To create an asset (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose **Create asset**.
4. On the **Create asset** page, do the following:

   a. For **Model**, choose the asset model from which to create an asset.

   > **Note**
   >
   > If your model isn't **ACTIVE**, you must wait until it's active, or resolve issues if it's **FAILED**.

   b. Enter a **Name** for your asset.

   c. (Optional) Add tags for your asset. For more information, see Tagging your AWS IoT SiteWise resources.

   d. Choose **Create asset**.

   When you create an asset, the AWS IoT SiteWise console navigates to the new asset's page. On this page, you can see the asset's **Status**, which is initially **CREATING**. This page automatically updates, so you can wait for the asset's status to update.

> **Note**
>
> The asset creation process can take up to a minute. After the **Status** is **ACTIVE**, you can perform update operations on your asset. For more information, see Asset and model states.

After you create an asset, see Configuring a new asset.

Creating an asset (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to create an asset from an asset model.

You must have an `assetModelId` to create an asset. If you created an asset model, but don't know its `assetModelId`, use the ListAssetModels API to view all of your asset models.

To create an asset from an asset model, use the CreateAsset API with the following parameters:

- `assetName` – The new asset's name. Give your asset a name to help you identify it.
- `assetModelId` – The ID of the asset. This is the actual ID in UUID format, or the `externalId:myExternalId` if it has one. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

To create an asset (AWS CLI)

- Run the following command to create an asset. Replace *`asset-name`* with a name for the asset and *`asset-model-id`* with the ID or the external ID of the asset model.

```
aws iotsitewise create-asset \
  --asset-name asset-name \
  --asset-model-id asset-model-id
```

  The operation returns a response that contains your new asset's details and status in the following format.

```
{
  "assetId": "String",
  "assetArn": "String",
```

```
  "assetStatus": {
    "state": "String",
    "error": {
      "code": "String",
      "message": "String"
    }
  }
}
```

The asset's `state` is `CREATING` until the asset creates.

> **Note**
>
> The asset creation process can take up to a minute. To check your asset's status, use the DescribeAsset operation with your asset's ID as the `assetId` parameter. After the asset's `state` is `ACTIVE`, you can perform update operations on your asset. For more information, see Asset and model states.

After you create an asset, see Configuring a new asset.

Configuring a new asset

Finish configuring your asset with any of the following optional actions:

- Mapping industrial data streams to asset properties if your asset has measurement properties.
- Updating attribute values if your asset has unique attribute values.
- Associating and disassociating assets if your asset is a parent asset.

Searching assets

Use the AWS IoT SiteWise console search functionality to find assets based on metadata and real-time property value filters.

Prerequisites

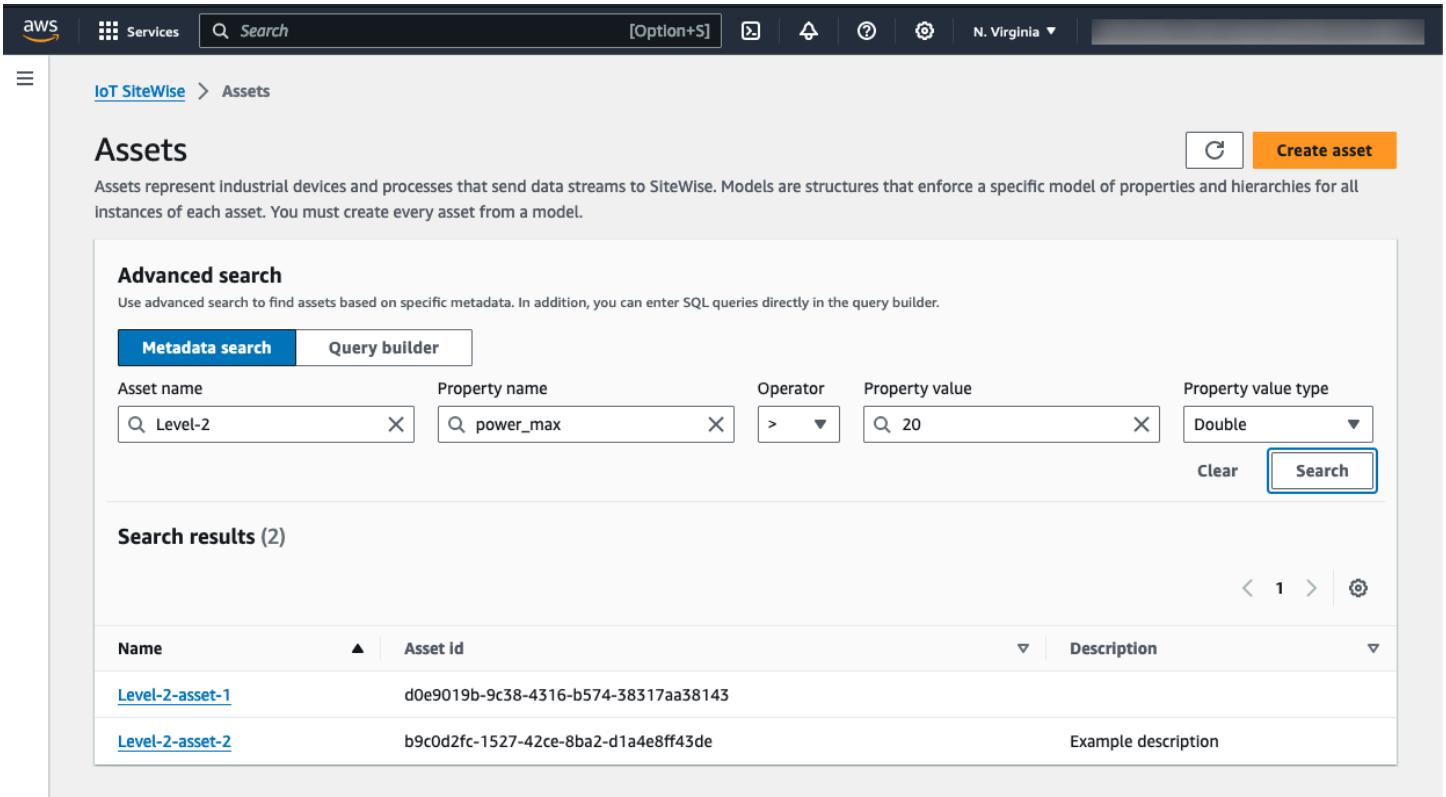
AWS IoT SiteWise requires permissions to integrate with AWS IoT TwinMaker to better organize, and model industrial data. If you have granted permissions to AWS IoT SiteWise, use the

ExecuteQuery API. If you have not granted permissions to AWS IoT SiteWise, and need assistance getting started, see Integrating AWS IoT SiteWise and AWS IoT TwinMaker.

Advanced search on AWS IoT SiteWise console

Metadata search

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Advanced search** under **Assets**.
3. Under **Advanced search** choose the **Metadata search** option.
4. Fill in the parameters. Fill in as many fields as possible for an efficient search.
   a. **Asset name** — Enter a full asset name, or a partial name for a wide search.
   b. **Property name** — Enter a full property name, or a partial name for a wide search.
   c. **Operator** — Choose an operator from:
      - **=**
      - **<**
      - **>**
      - **<=**
      - **>=**
   d. **Property value** — This value is compared with the property's latest value.
   e. **Property value type** — The data type of the property. Choose from the following:
      - **Double**
      - **Integer**
      - **String**
      - **Boolean**
5. Choose **Search**.
6. From the **Search results** table, choose the asset from the **Name** column. This takes you to the detailed asset page for that asset.



Partial search

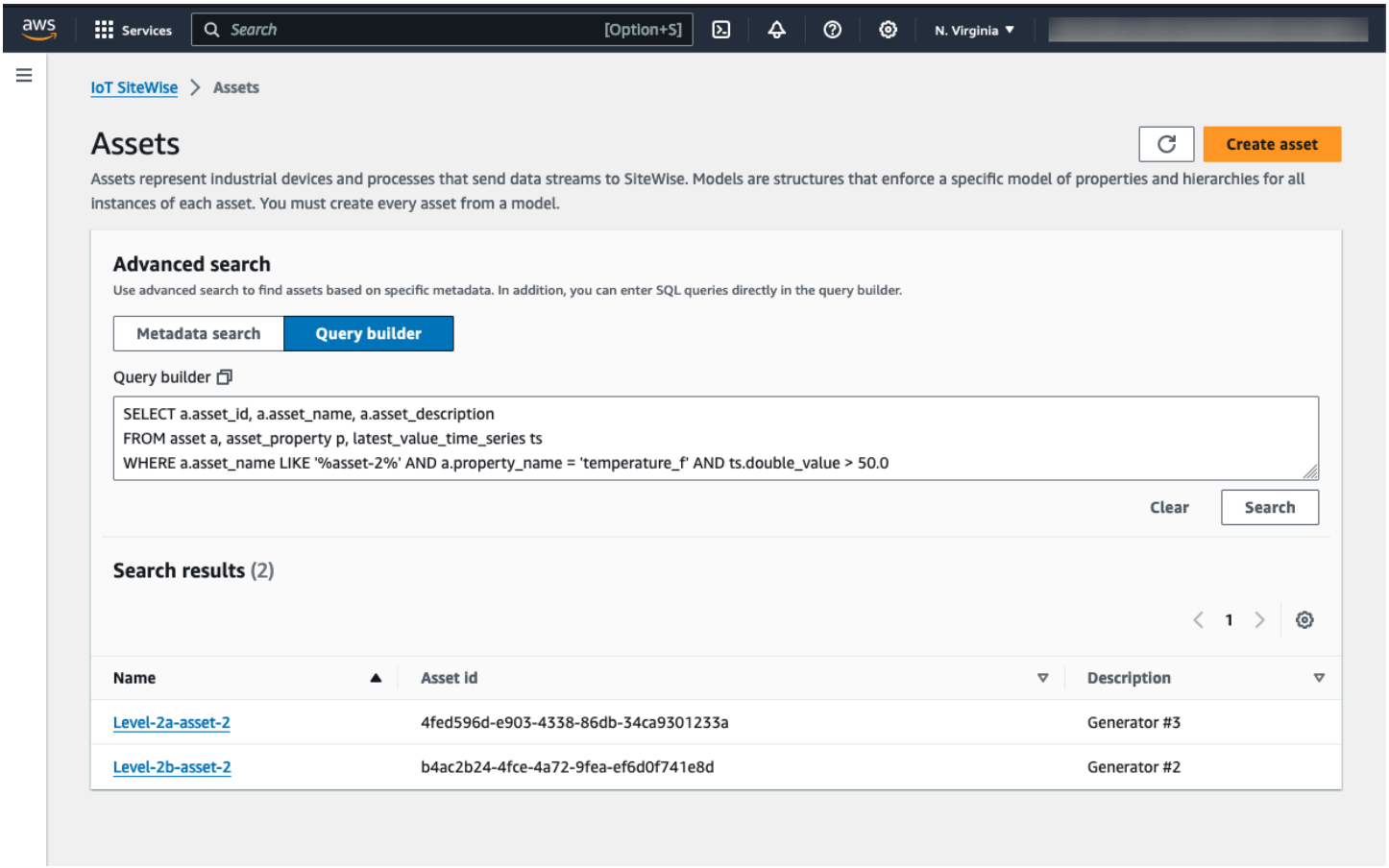
All parameters do not need to be provided for an asset search. Here are some examples of partial searches using the Metadata search option:

- Find assets by their name:
  - Enter a value in the **Asset name** field alone.
  - The **Property name** and **Property value** fields are empty.
- Find assets containing properties with a specific name:
  - Enter a value in the **Property name** field alone.
  - The **Asset name** and **Property value** fields are empty.
- Find assets based on the latest values of their properties:
  - Enter values in the **Property name** and **Property value** fields.
  - Select an **Operator** and **Property value type**.

Query builder search

1. Navigate to the AWS IoT SiteWise console.

2. In the navigation pane, choose **Advanced search** under **Assets**.
3. Under **Advanced search** choose the **Query builder** option.
4. In the **Query builder** pane, write your SQL query to retrieve an `asset_name`, `asset_id` and `asset_description`.
5. Choose **Search**.
6. From the **Search results** table, choose the asset from the **Name** column. This takes you to the detailed asset page for that asset.



Note

- The `SELECT` clause in the SQL query must include the `asset_name` and `asset_id` fields to ensure a valid asset in the **Search results** table.
- The **Query builder** only displays the **Name**, **Asset id**, and **Description** in the results table. Adding more fields to the `SELECT` clause does not add more columns to the results table

Mapping industrial data streams to asset properties

You can define a property alias on asset property. This helps you identify an asset property when you ingest or retrieve asset data. If your asset has measurement properties, you can define the property aliases to map your data streams to those measurement properties.

This process requires that you know your property alias.

- If you ingest data from OPC-UA servers using an OPC-UA data source in a SiteWise Edge gateway, your property alias is the path to a variable under the **Objects** node, starting with `/`.

  Example

  If the path to your variable is `company/windfarm/3/turbine/7/temperature`, then your property alias is `/company/windfarm/3/turbine/7/temperature`.

  For more information about OPC-UA information architecture, see Information Model and Address Spacing mapping in the *OPC UA Online Reference*.

  > **Notes**
  >
  > - If you configure a data stream prefix for your OPC-UA source, you must include that prefix in the property alias for all data streams from that source.
  >
  >   **Example**
  >
  >   If `/RentonWA` is a prefix, then the previous alias is `/RentonWA/company/windfarm/3/turbine/7/temperature`.
  > - Property aliases can contain up to 1,000 bytes. OPC-UA variables paths can contain up to 4,096 bytes. Currently, AWS IoT SiteWise doesn't support ingesting data from OPC-UA variables with long paths.

- If you ingest data from Modbus servers using a Modbus TCP data source in a SiteWise Edge gateway, your property alias is:

  Modbus `register set tag name`

  Use this value to send data from this register set to an asset property.
- If you ingest data from other sources, such as using AWS IoT rules or the API, you must define your property aliases. You can define a property alias naming system that is applicable to your

device configuration. For example, if you ingest data from AWS IoT things, you can include the thing name in property aliases to uniquely identify data streams. For more information about this example, see the Ingesting data from AWS IoT things tutorial.

Property aliases must be unique within a Region and AWS account. AWS IoT SiteWise returns an error if you set a property alias to one that already exists on another asset property.

If you have multiple OPC-UA sources with identical data stream paths, add a prefix to each source's paths to form unique aliases. For more information, see Configuring data sources.

> **Note**
>
> This section describes how to set property aliases for measurement properties. For more information about how to set property aliases for external alarm state properties, see Mapping external alarm state streams.

Topics

- Setting a property alias (console)
- Setting a property alias (AWS CLI)

Setting a property alias (console)

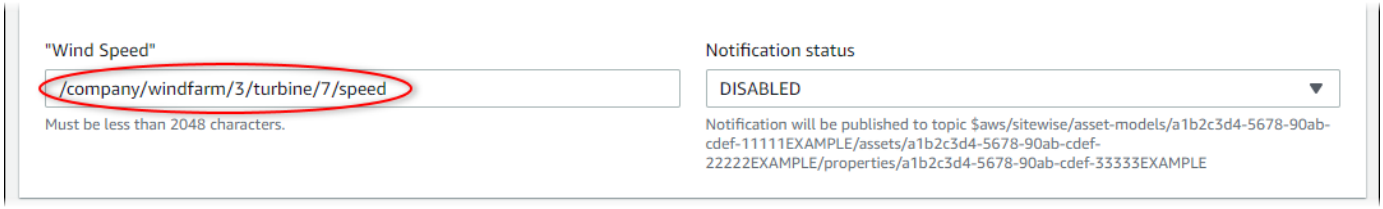
You can use the AWS IoT SiteWise console to set an alias for an asset property.

To set a property alias (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the asset for which you want to set a property alias.

   > **Tip**
   >
   > You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. Choose **Edit**.
5. Find the property for which you want to set an alias, and then enter the property alias.



6. Choose **Save**.

Setting a property alias (AWS CLI)

Use the AWS Command Line Interface (AWS CLI) to set an alias for an asset property.

You must know your asset's `assetId` and property's `propertyId` to complete this procedure. You can also use the external ID. If you created an asset and don't know its `assetId`, use the ListAssets API to list all the assets for a specific model. Use the DescribeAsset operation to view your asset's properties including property IDs.

Use the UpdateAssetProperty operation, to map a data stream to your asset's property. Specify the following parameters:

- `assetId` – The asset's ID or external ID. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.
- `propertyId` – The asset property's ID or external ID.
- `propertyAlias` – The data stream's path to alias to the property.
- `propertyNotificationState` – The property value notification state: `ENABLED` or `DISABLED`. Specify the property's existing notification state when you update the property alias. You can retrieve the existing notification state with the DescribeAssetProperty operation.

  If you omit this parameter, the new notification state is `DISABLED`. For more information about property notifications, see Interacting with other AWS services.

To set a property alias (AWS CLI)

1. Run the following command to retrieve the property's current notification state. Replace *`asset-id`* and *`property-id`* with the asset property's IDs.

```
aws iotsitewise describe-asset-property \
  --asset-id asset-id \
```

```
  --property-id property-id
```

The operation returns a response that contains the asset property's details in the following format. The property notification state is in `assetProperty.notification.state` in the JSON object.

```
{
  "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
  "assetName": "Wind Turbine 7",
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
  "assetProperty": {
    "id": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
    "name": "Wind Speed",
    "notification": {
      "topic": "$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/
assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE",
      "state": "ENABLED"
    },
    "dataType": "DOUBLE",
    "unit": "m/s",
    "type": {
      "measurement": {}
    }
  }
}
```

2. Run the following command to set the asset property's alias. Replace *property-alias* with the property alias and *notification-state* with the notification state, or omit `--property-notification-state` to disable notifications. You can optionally update the asset's unit with a new *unit* and `--property-unit`.

```
aws iotsitewise update-asset-property \
  --asset-id asset-id \
  --property-id property-id \
  --property-alias property-alias \
  --property-notification-state notification-state \
  --property-unit unit
```

3. To verify the alias has been set, run the following command to retrieve the property's details. Replace *asset-id* and *property-id* with the asset property's IDs.

```
aws iotsitewise describe-asset-property \
  --asset-id asset-id \
  --property-id property-id
```

The operation returns a response that contains the asset property's details in the following format. The property alias is `assetProperty.alias` in the JSON object and is set to `myAlias` in this example.

```
{
  "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
  "assetName": "Wind Turbine 7",
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
  "assetProperty": {
    "alias": "myAlias",
    "id": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
    "name": "Wind Speed",
    "notification": {
      "topic": "$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/
assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE",
      "state": "ENABLED"
    },
    "dataType": "DOUBLE",
    "unit": "m/s",
    "type": {
      "measurement": {}
    }
  }
}
```

Updating attribute values

Assets inherit the attributes of their asset model, including the default value of the attribute. In some cases, you will want to keep the asset model's default attribute, such as for an asset manufacturer property. In other cases, you will want to update the inherited attribute, such as for an asset's latitude and longitude.

Updating an attribute value (console)

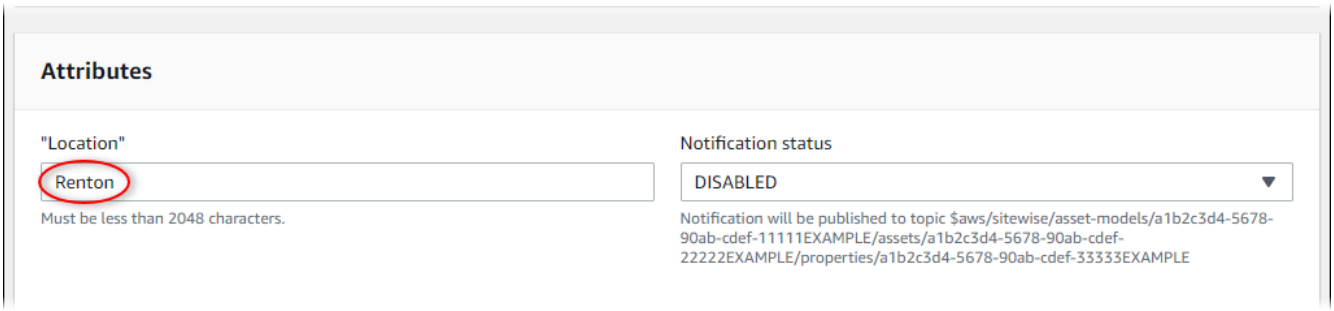
You can use the AWS IoT SiteWise console to update the value of an attribute asset property.

To update an attribute's value (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the asset for which you want to update an attribute.

> **Tip**
>
> You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. Choose **Edit**.
5. Find the attribute to update, and then enter its new value.



6. Choose **Save**.

Updating an attribute value (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to update an attribute value.

You must know your asset's `assetId` and property's `propertyId` to complete this procedure. You can also use the external ID. If you created an asset and don't know its `assetId`, use the ListAssets API to list all the assets for a specific model. Use the DescribeAsset operation to view your asset's properties including property IDs.

Use the BatchPutAssetPropertyValue operation to assign attribute values to your asset. You can use this operation to set multiple attributes at once. This operation's payload contains a list of entries, and each entry contains the asset ID, property ID, and attribute value.

To update an attribute's value (AWS CLI)

1. Create a file called `batch-put-payload.json` and copy the following JSON object into the file. This example payload demonstrates how to set a wind turbine's latitude and longitude. Update the IDs, values, and timestamps to modify the payload for your use case.

```
{
  "entries": [
    {
      "entryId": "windfarm3-turbine7-latitude",
      "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "propertyId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
      "propertyValues": [
        {
          "value": {
            "doubleValue": 47.6204
          },
          "timestamp": {
            "timeInSeconds": 1575691200
          }
        }
      ]
    },
    {
      "entryId": "windfarm3-turbine7-longitude",
      "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "propertyId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE",
      "propertyValues": [
        {
          "value": {
            "doubleValue": 122.3491
          },
          "timestamp": {
            "timeInSeconds": 1575691200
          }
        }
      ]
    }
  ]
}
```

- Each entry in the payload contains an `entryId` that you can define as any unique string. If any request entries fail, each error will contain the `entryId` of the corresponding request so that you know which requests to retry.
- To set an attribute value, you can include one timestamp-quality-value (TQV) structure in the list of `propertyValues` for each attribute property. This structure must contain the new `value` and the current `timestamp`.
  - `value` – A structure that contains one of the following fields, depending on the type of the property being set:
    - `booleanValue`
    - `doubleValue`
    - `integerValue`
    - `stringValue`
  - `timestamp` – A structure that contains the current Unix epoch time in seconds, `timeInSeconds`. AWS IoT SiteWise rejects any data points with timestamps that existed longer than 7 days in the past or newer than 5 minutes in the future.

  For more information about how to prepare a payload for BatchPutAssetPropertyValue, see Ingesting data using the AWS IoT SiteWise API.

2. Run the following command to send the attribute values to AWS IoT SiteWise:

```
aws iotsitewise batch-put-asset-property-value -\-cli-input-json file://batch-put-payload.json
```

Associating and disassociating assets

If your asset's model defines any child asset model hierarchies, you can associate child assets to your asset. Parent assets can access and aggregate data from associated assets. For more information about hierarchical asset models, see Defining asset model hierarchies.

Topics

- Associating and disassociating assets (console)
- Associating and disassociating assets (AWS CLI)

Associating and disassociating assets (console)

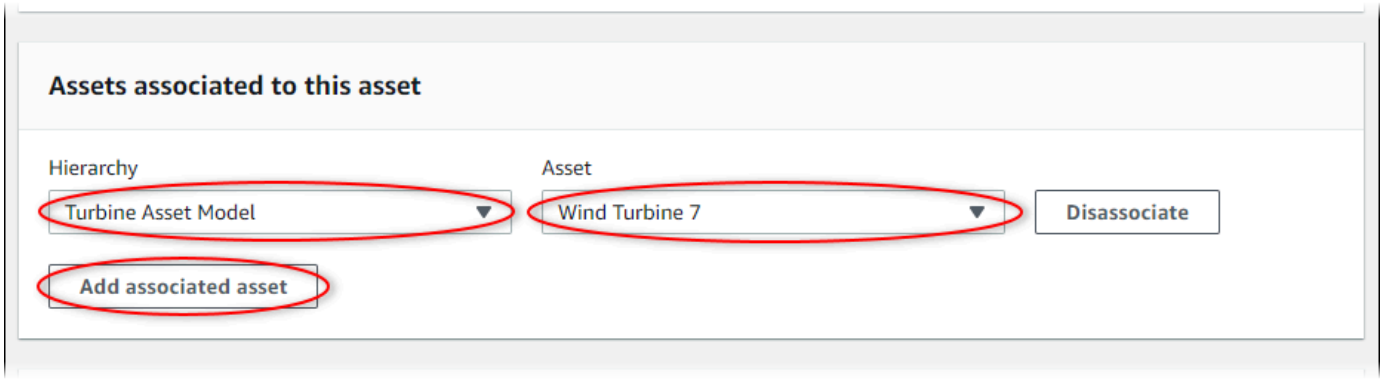
You can use the AWS IoT SiteWise console to associate and disassociate assets.

To associate an asset (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the parent asset for which you want to associate a child asset.

> **Tip**
>
> You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. Choose **Edit**.
5. In **Assets associated to this asset**, choose **Add associated asset**.



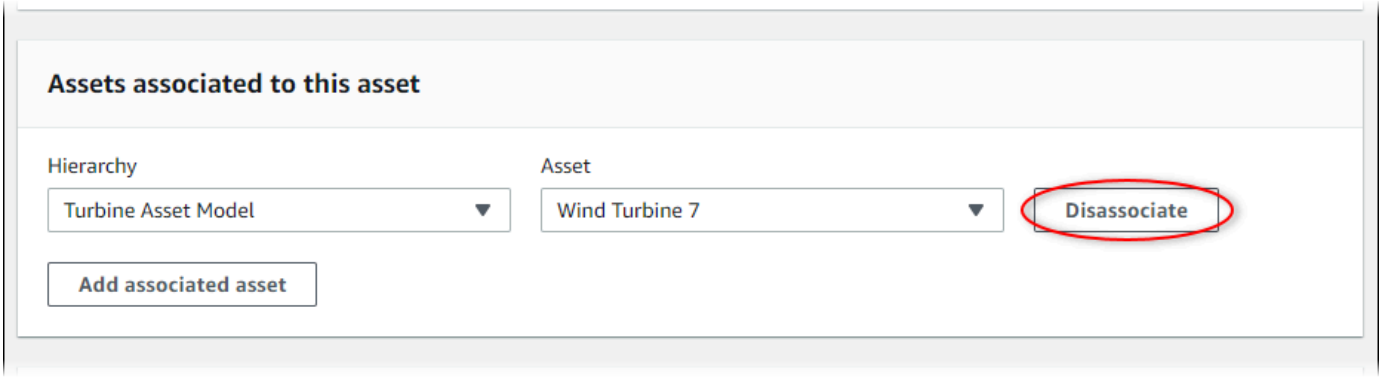
6. For **Hierarchy**, choose the hierarchy that defines the relationship between the parent asset and the child asset.
7. For **Asset**, choose the child asset to associate.
8. Choose **Save**.

To disassociate an asset (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the parent asset for which you want to disassociate a child asset.

> **Tip**
>
> You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. Choose **Edit**.
5. In **Assets associated to this asset**, choose **Disassociate** for the asset.



6. Choose **Save**.

Associating and disassociating assets (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to associate and disassociate assets.

For this procedure, you must know the ID of the hierarchy (`hierarchyId`) in the parent asset model that defines the relationship to the child asset model. Use the DescribeAsset operation to find the hierarchy ID in the response.

To find a hierarchy ID

- Run the following command to describe the parent asset. Replace *`parent-asset-id`* with the parent asset's ID or external ID.

  ```
  aws iotsitewise describe-asset --asset-id parent-asset-id
  ```

  The operation returns a response that contains the asset's details. The response contains an `assetHierarchies` list that has the following structure:

  ```
  {
    ...
    "assetHierarchies": [
  ```

```
    {
      "id": "String",
      "name": "String"
    }
  ],
  ...
}
```

The hierarchy ID is the `id` value for a hierarchy in the list of asset hierarchies.

After you have the hierarchy ID, you can associate or disassociate an asset with that hierarchy.

To associate a child asset to a parent asset, use the AssociateAssets operation. To disassociate a child asset from a parent asset, use the DisassociateAssets operation. Specify the following parameters, which are the same for both operations:

- `assetId` – The parent asset's ID or external ID.
- `hierarchyId` – The hierarchy ID or external ID in the parent asset.
- `childAssetId` – The child asset's ID or external ID.

To associate an asset (AWS CLI)

- Run the following command to associate a child asset to a parent asset. Replace *parent-asset-id*, *hierarchy-id*, and *child-asset-id* with the respective IDs:

```
aws iotsitewise associate-assets \
  --asset-id parent-asset-id \
  --hierarchy-id hierarchy-id \
  --child-asset-id child-asset-id
```

To disassociate an asset (AWS CLI)

- Run the following command to disassociate a child asset from a parent asset. Replace *parent-asset-id*, *hierarchy-id*, and *child-asset-id* with the respective IDs:

```
aws iotsitewise disassociate-assets \
  --asset-id parent-asset-id \
  --hierarchy-id hierarchy-id \
```

```
  --child-asset-id child-asset-id
```

Updating assets and models

You can update your assets, asset models, and component models in AWS IoT SiteWise to modify their names and definitions. These update operations are asynchronous and take time to propagate through AWS IoT SiteWise. Check the status of the asset or model before you make additional changes. You must wait until the changes propagate before you can continue to use the updated asset or model.

Topics

Updating assets

You can use the AWS IoT SiteWise console or API to update an asset's name.

When you update an asset, the asset's status is `UPDATING` until the changes propagate. For more information, see Asset and model states.

Topics

Updating an asset (console)

You can use the AWS IoT SiteWise console to update asset details.

To update an asset (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the asset to update.

> **Tip**
>
> You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. Choose **Edit**.
5. Update the asset's **Name**.
6. (Optional) On this page, update other information for the asset. For more information, see the following:
   - Mapping industrial data streams to asset properties
   - Updating attribute values
   - Interacting with other AWS services
7. Choose **Save**.

Updating an asset (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to update an asset's name.

Use the UpdateAsset operation to update an asset. Specify the following parameters:

- `assetId` – The ID of the asset. This is the actual ID in UUID format, or the `externalId:myExternalId` if it has one. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.
- `assetName` – The asset's new name.

To update an asset's name (AWS CLI)

- Run the following command to update an asset's name. Replace *`asset-id`* with the ID or external ID of the asset. Update the *`asset-name`* with the new name for the asset.

```
aws iotsitewise update-asset \
  --asset-id asset-id \
  --asset-name asset-name
```

Updating asset models and component models

You can use the AWS IoT SiteWise console or API to update an asset model or component model.

You can't change the type or data type of an existing property, or the window of an existing metric. You also can't change the type of the model from asset model to component model, or the other way around.

Important

- If you remove a property from an asset model or component model, AWS IoT SiteWise deletes all previous data for that property. For component models, this affects **all asset models using that component model**, so be especially careful to understand how widely your change may apply.
- If you remove a hierarchy definition from an asset model, AWS IoT SiteWise disassociates all assets in that hierarchy.

When you update an asset model, every asset based on that model reflects any changes that you make to the underlying model. Until the changes propagate, each asset has the `UPDATING` state. You must wait until those assets return to the `ACTIVE` state before you interact with them. During this time, the updated asset model's status will be `PROPAGATING`.

When you update a component model, every asset model that incorporates that component model reflects the changes. Until the component model changes propagate, each affected asset model has the `UPDATING` state, followed by `PROPAGATING` as it updates its associated assets, as described in the preceding paragraph. You must wait until those asset models return to the `ACTIVE` state before you interact with them. During this time, the updated component model's status will be `PROPAGATING`.

For more information, see Asset and model states.

Topics

Updating an asset or component model (console)

You can use the AWS IoT SiteWise console to update an asset model or component model.

To update an asset model or component model (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Models**.
3. Choose the asset model or component model to update.
4. Choose **Edit**.
5. On the **Edit model** page, do any of the following:
   - In **Model details**, change the **Name** of the model.
   - Change any of the **Attribute definitions**. You can't change the **Data type** of existing attributes. For more information, see Defining static data (attributes).
   - Change any of the **Measurement definitions**. You can't change the **Data type** of existing measurements. For more information, see Defining data streams from equipment (measurements).
   - Change any of the **Transform definitions**. For more information, see Transforming data (transforms).
   - Change any of the **Metric definitions**. You can't change the **Time interval** of existing metrics. For more information, see Aggregating data from properties and other assets (metrics).
   - (Asset models only) Change any of the **Hierarchy definitions**. You can't change the **Hierarchy model** of existing hierarchies. For more information, see Defining asset model hierarchies.
6. Choose **Save**.

Updating an asset or component model (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to update an asset model or component model.

Use the UpdateAssetModel API to update the name, description, and properties of an asset model or component model. For asset models only, you can update hierarchies. Specify the following parameters:

- `assetModelId` – The ID of the asset. This is the actual ID in UUID format, or the `externalId:myExternalId` if it has one. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

Specify the updated model in the payload. To learn about the expected format of an asset model or component model, see Creating asset models.

> **Warning**
>
> The UpdateAssetModel API overwrites the existing model with the model that you provide in the payload. To avoid deleting your model's properties or hierarchies, you must include their IDs and definitions in the updated model payload. To learn how to query your model's existing structure, see the DescribeAssetModel operation.

> **Note**
>
> The following procedure can only update composite models of type `AWS/ALARM`. If you want to update `CUSTOM` composite models, use UpdateAssetModelCompositeModel instead. For more information, see Updating custom composite models (Components).

To update an asset model or component model (AWS CLI)

1. Run the following command to retrieve the existing model definition. Replace *`asset-model-id`* with the ID or the external ID of the asset model or component model to update.

   ```
   aws iotsitewise describe-asset-model --asset-model-id asset-model-id
   ```

   The operation returns a response that contains the model's details. The response has the following structure.

   ```
   {
     "assetModelId": "String",
     "assetModelArn": "String",
     "assetModelName": "String",
     "assetModelDescription": "String",
     "assetModelProperties": Array of AssetModelProperty,
   ```

```
  "assetModelHierarchies": Array of AssetModelHierarchyDefinition,
  "assetModelCompositeModels": Array of AssetModelCompositeModel,
  "assetModelCompositeModelSummaries": Array of AssetModelCompositeModelSummary,
  "assetModelCreationDate": "String",
  "assetModelLastUpdateDate": "String",
  "assetModelStatus": {
    "state": "String",
    "error": {
      "code": "String",
      "message": "String"
    },
  "assetModelType": "String"
  }
}
```

For more information, see the DescribeAssetModel operation.

2. Create a file called `update-asset-model.json` and copy the previous command's response into the file.
3. Remove the following key-value pairs from the JSON object in `update-asset-model.json`:

   - `assetModelId`
   - `assetModelArn`
   - `assetModelCompositeModelSummaries`
   - `assetModelCreationDate`
   - `assetModelLastUpdateDate`
   - `assetModelStatus`
   - `assetModelType`

   The UpdateAssetModel operation expects a payload with the following structure:

```
{
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
  "assetModelHierarchies": Array of AssetModelHierarchyDefinition,
  "assetModelCompositeModels": Array of AssetModelCompositeModel
}
```

4. In `update-asset-model.json`, do any of the following:
   - Change the asset model's name (`assetModelName`).
   - Change, add, or remove the asset model's description (`assetModelDescription`).
   - Change, add, or remove any of the asset model's properties (`assetModelProperties`). You can't change the `dataType` of existing properties or the `window` of existing metrics. For more information, see Defining data properties.
   - Change, add, or remove any of the asset model's hierarchies (`assetModelHierarchies`). You can't change the `childAssetModelId` of existing hierarchies. For more information, see Defining asset model hierarchies.
   - Change, add, or remove any of the asset model's composite models of type `AWS/ALARM` (`assetModelCompositeModels`). Alarms monitor other properties so that you can identify when equipment or processes require attention. Each alarm definition is a composite model that standardizes the set of properties that the alarm uses. For more information, see Monitoring data with alarms and Defining alarms on asset models.
5. Run the following command to update the asset model with the definition stored in `update-asset-model.json`. Replace *`asset-model-id`* with the ID of the asset model:

```
aws iotsitewise update-asset-model \
  --asset-model-id asset-model-id \
  --cli-input-json file://model-payload.json
```

Updating custom composite models (Components)

You can use the AWS IoT SiteWise API to update a custom composite model, or the AWS IoT SiteWise console to update components.

Topics

- Updating a component (console)
- Updating a custom composite model (AWS CLI)

Updating a component (console)

You can use the AWS IoT SiteWise console to update a component.

To update a component (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Models**.
3. Choose the asset model where the component is.
4. On the **Properties** tab, choose **Components**.
5. Choose the component that you want to update.
6. Choose **Edit**.
7. On the **Edit component** page, do any of the following:
   - In **Model details**, change the **Name** of the model.
   - Change any of the **Attribute definitions**. You can't change the **Data type** of existing attributes. For more information, see Defining static data (attributes).
   - Change any of the **Measurement definitions**. You can't change the **Data type** of existing measurements. For more information, see Defining data streams from equipment (measurements).
   - Change any of the **Transform definitions**. For more information, see Transforming data (transforms).
   - Change any of the **Metric definitions**. You can't change the **Time interval** of existing metrics. For more information, see Aggregating data from properties and other assets (metrics).
8. Choose **Save**.

Updating a custom composite model (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to update a custom composite model.

To update the name or description, use the UpdateAssetModelCompositeModel operation. For inline custom composite models only, you can also update the properties. You can't update the properties of a component-model-based custom composite model, because its referenced component model provides its associated properties.

Important

If you remove a property from a custom composite model, AWS IoT SiteWise deletes all previous data for that property. You can't change the type or data type of an existing property.
To replace an existing composite model property with a new one with the same `name`, do the following:

1. Submit an `UpdateAssetModelCompositeModel` request with the entire existing property removed.
2. Submit a second `UpdateAssetModelCompositeModel` request that includes the new property. The new asset property will have the same `name` as the previous one and AWS IoT SiteWise will generate a new unique `id`.

To update a custom composite model (AWS CLI)

1. To retrieve the existing composite model definition, run the following command. Replace *`composite-model-id`* with the ID or the external ID of the custom composite model to update, and *`asset-model-id`* with the asset model that the custom composite model is associated with. For more information, see in the *AWS IoT SiteWise User Guide*.

   ```
   aws iotsitewise describe-asset-model-composite-model \
   --asset-model-composite-model-id composite-model-id \
   --asset-model-id asset-model-id
   ```

   For more information, see the DescribeAssetModelCompositeModel operation.
2. Create a file called `update-custom-composite-model.json`, and then copy the previous command's response into the file.
3. Remove every key-value pair from the JSON object in `update-custom-composite-model.json` except for the following fields:

   - `assetModelCompositeModelName`
   - `assetModelCompositeModelDescription` (if present)
   - `assetModelCompositeModelProperties` (if present)
4. In `update-custom-composite-model.json`, do any of the following:

- Change the value of `assetModelCompositeModelName`.
- Add or remove `assetModelCompositeModelDescription`, or change its value.
- For inline custom composite models only: Change, add, or remove any of the asset model's properties in `assetModelCompositeModelProperties`.

For more information about the required format for this file, see the request syntax for UpdateAssetModelCompositeModel.

5. Run the following command to update the custom composite model with the definition stored in `update-custom-composite-model.json`. Replace *`composite-model-id`* with the ID of the composite model, and *`asset-model-id`* with the ID of the asset model it's in.

```
aws iotsitewise update-asset-model-composite-model \
--asset-model-composite-model-id composite-model-id \
--asset-model-id asset-model-id \
--cli-input-json file://update-custom-composite-model.json
```

Deleting assets and models

You can delete your assets and models from AWS IoT SiteWise when you're done with them. The delete operations are asynchronous and take time to propagate through AWS IoT SiteWise.

Topics

- Deleting assets
- Deleting asset models

Deleting assets

You can use the AWS IoT SiteWise console or API to delete an asset.

Before you can delete an asset, you must first disassociate its child assets and disassociate it from its parent asset. For more information, see Associating and disassociating assets. If you use the AWS Command Line Interface (AWS CLI), you can use the ListAssociatedAssets operation to list an asset's children.

When you delete an asset, its status is `DELETING` until the changes propagate. For more information, see Asset and model states. After the asset is deleted, you can't query that asset. If you do, the API returns an HTTP 404 response.

> **Important**
>
> AWS IoT SiteWise deletes all property data for deleted assets.

Topics

Deleting an asset (console)

You can use the AWS IoT SiteWise console to delete an asset.

To delete an asset (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the asset to delete.

   > **Tip**
   >
   > You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. If the asset has any **Associated assets**, delete each asset. You can choose an asset's name to navigate to its page, where you can delete it.
5. On the asset's page, choose **Delete**.
6. In the **Delete asset** dialog box, do the following:

   a. Enter **`Delete`** to confirm deletion.

   b. Choose **Delete**.

Deleting an asset (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to delete an asset.

Use the DeleteAsset operation to delete an asset. Specify the following parameter:

- `assetId` – The ID of the asset. This is the actual ID in UUID format, or the `externalId:myExternalId` if it has one. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

To delete an asset (AWS CLI)

1. Run the following command to list the asset's hierarchies. Replace *asset-id* with the ID or the external ID of the asset:

   ```
   aws iotsitewise describe-asset --asset-id asset-id
   ```

   The operation returns a response that contains the asset's details. The response contains an `assetHierarchies` list that has the following structure:

   ```
   {
     ...
     "assetHierarchies": [
       {
         "id": "String",
         "name": "String"
       }
     ],
     ...
   }
   ```

   For more information, see the DescribeAsset operation.

2. For each hierarchy, run the following command to list the asset's children that are associated with that hierarchy. Replace *asset-id* with the ID or external ID of the asset and *hierarchy-id* with the ID or external ID of the hierarchy.

   ```
   aws iotsitewise list-associated-assets \
     --asset-id asset-id \
     --hierarchy-id hierarchy-id
   ```

For more information, see the ListAssociatedAssets operation.

3. Run the following command to delete each associated asset and then to delete the asset. Replace *asset-id* with the ID or external ID of the asset.

```
aws iotsitewise delete-asset --asset-id asset-id
```

Deleting asset models

You can use the AWS IoT SiteWise console or API to delete an asset model.

Before you can delete an asset model, you must first delete all assets that were created from the asset model.

When you delete an asset model, its status is `DELETING` until the changes propagate. For more information, see Asset and model states. After the asset model is deleted, you can't query that asset model. If you do, the API returns an HTTP 404 response.

Topics

- Deleting an asset model (console)
- Deleting an asset model (AWS CLI)

Deleting an asset model (console)

You can use the AWS IoT SiteWise console to delete an asset model.

To delete an asset model (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Models**.
3. Choose the asset model to delete.
4. If the model has any **Assets**, delete each asset. Choose an asset's name to navigate to its page, where you can delete it. For more information, see Deleting an asset (console).
5. On the model's page, choose **Delete**.
6. In the **Delete model** dialog box, do the following:

a. Enter **`Delete`** to confirm deletion.

b. Choose **Delete**.

Deleting an asset model (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to delete an asset model.

Use the DeleteAssetModel operation to delete an asset model. Specify the following parameter:

- `assetModelId` – The ID of the asset. This is the actual ID in UUID format, or the `externalId:myExternalId` if it has one. For more information, see Referencing objects with external IDs in the *AWS IoT SiteWise User Guide*.

To delete an asset model (AWS CLI)

1. Run the following command to list all assets created from the model. Replace *`asset-model-id`* with the ID or the external ID of the asset model.

   ```
   aws iotsitewise list-assets --asset-model-id asset-model-id
   ```

   For more information, see the ListAssets operation.
2. If the previous command returns any assets from the model, delete each asset. For more information, see Deleting an asset (AWS CLI).
3. Run the following command to delete the asset model. Replace *`asset-model-id`* with the ID or external ID of the asset model.

   ```
   aws iotsitewise delete-asset-model --asset-model-id asset-model-id
   ```

Bulk operations with assets and models

To work with a large number of assets or asset models, use bulk operations to bulk import and export resources to a different location. For example, you can create a data file that defines assets or asset models in an Amazon S3 bucket, and use bulk import to create or update them in AWS IoT SiteWise. Alternatively, if you have a large number of assets or asset models in AWS IoT SiteWise, you can export them to Amazon S3.

> **Note**
>
> You perform bulk operations in AWS IoT SiteWise by calling operations in the AWS IoT TwinMaker API. You can do this without setting up AWS IoT TwinMaker or creating an AWS IoT TwinMaker workspace. All you need is an Amazon S3 bucket where you can place your AWS IoT SiteWise content.

Topics

Key concepts and terminology

AWS IoT SiteWise bulk import and export features rely on the following concepts and terminology:

- **Import**: The action of moving assets or asset models from a file in an Amazon S3 bucket to AWS IoT SiteWise.
- **Export**: The action of moving assets or asset models from AWS IoT SiteWise to an Amazon S3 bucket.
- **Source**: The starting location of where you want to move content from.

  For example, an Amazon S3 bucket is an import source, and AWS IoT SiteWise is an export source.
- **Destination**: The desired location of where you want to move your content to.

  For example, an Amazon S3 bucket is an export destination, and AWS IoT SiteWise is an import destination.

- **AWS IoT SiteWise Schema**: This schema is used to import and export metadata from AWS IoT SiteWise.
- **Top-level resource:** An AWS IoT SiteWise resource that you can individually create or update, such as an asset or asset model.
- **Sub-resource:** A nested AWS IoT SiteWise resource within a top-level resource. Examples include properties, hierarchies, and composite models.
- **Metadata**: Key information required to import or export resources successfully. Examples of metadata are definitions of assets and asset models.
- **metadataTransferJob**: The object created when you run `CreateMetadataTransferJob`.

Supported functionality

This topic explains what you can do when you run a bulk operation. Bulk operations support the following functionality:

- **Top-level resource creation:** When you import an asset or asset model that doesn't define an ID, or whose ID doesn't match that of an existing one, then it will be created as a new resource.
- **Top-level resource replacement:** When you import an asset or asset model whose ID matches one that already exists, then it will replace the existing resource.
- **Subresource creation, replacement, or deletion:** When your import replaces a top-level resource such as an asset or asset model, then the new definition replaces all sub-resources, such as properties, hierarchies, or composite models.

  For example, if you update an asset model during a bulk import, and the updated version defines a property that wasn't present on the original, then a new property is created. If it defines a property that already exists, then the existing property will be updated. If the updated asset model omits a property that was present on the original, then the property is deleted.
- **No top-level resource deletion:** Bulk operations don't delete an asset or asset model. Bulk operations only create or update them.

Bulk operation prerequisites

This section explains bulk operation prerequisites, including AWS Identity and Access Management (IAM) permissions for exchanging resources between AWS services and your local machine. Before you start a bulk operation, complete the following prerequisite:

- Create an Amazon S3 bucket to store resources. For more information about using Amazon S3, see What is Amazon S3?

IAM permissions

To perform bulk operations, you must create an AWS Identity and Access Management (IAM) policy with permissions that allow the exchange of AWS resources between Amazon S3, AWS IoT SiteWise, and your local machine. For more information about creating IAM policies, see Creating IAM policies.

To perform bulk operations, you need the following policies.

AWS IoT SiteWise policy

This policy allows access to the required AWS IoT SiteWise API actions for bulk operations:

```
{
    "Sid": "SiteWiseApiAccess",
    "Effect": "Allow",
    "Action": [
        "iotsitewise:CreateAsset",
        "iotsitewise:CreateAssetModel",
        "iotsitewise:UpdateAsset",
        "iotsitewise:UpdateAssetModel",
        "iotsitewise:UpdateAssetProperty",
        "iotsitewise:ListAssets",
        "iotsitewise:ListAssetModels",
        "iotsitewise:ListAssetProperties",
        "iotsitewise:ListAssetModelProperties",
        "iotsitewise:ListAssociatedAssets",
        "iotsitewise:DescribeAsset",
        "iotsitewise:DescribeAssetModel",
        "iotsitewise:DescribeAssetProperty",
        "iotsitewise:AssociateAssets",
        "iotsitewise:DisassociateAssets",
        "iotsitewise:AssociateTimeSeriesToAssetProperty",
        "iotsitewise:DisassociateTimeSeriesFromAssetProperty",
        "iotsitewise:BatchPutAssetPropertyValue",
        "iotsitewise:BatchGetAssetPropertyValue",
        "iotsitewise:TagResource",
        "iotsitewise:UntagResource",
        "iotsitewise:ListTagsForResource",
```

```
        "iotsitewise:CreateAssetModelCompositeModel",
        "iotsitewise:UpdateAssetModelCompositeModel",
        "iotsitewise:DescribeAssetModelCompositeModel",
        "iotsitewise:DeleteAssetModelCompositeModel",
        "iotsitewise:ListAssetModelCompositeModels",
        "iotsitewise:ListCompositionRelationships",
        "iotsitewise:DescribeAssetCompositeModel"
    ],
    "Resource": "*"
}
```

AWS IoT TwinMaker policy

This policy allows access to the AWS IoT TwinMaker API operations that you use to work with bulk operations:

```
{
    "Sid": "MetadataTransferJobApiAccess",
    "Effect": "Allow",
    "Action": [
        "iottwinmaker:CreateMetadataTransferJob",
        "iottwinmaker:CancelMetadataTransferJob",
        "iottwinmaker:GetMetadataTransferJob",
        "iottwinmaker:ListMetadataTransferJobs"
    ],
    "Resource": "*"
}
```

Amazon S3 policy

This policy provides access to Amazon S3 buckets for transferring metadata for bulk operations.

For a specific Amazon S3 bucket

If you use one specific bucket for working with your bulk operations metadata, this policy provides access to that bucket:

```
{
    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
```

```
        "s3:GetBucketLocation",
        "s3:ListBucket",
        "s3:AbortMultipartUpload",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts"
    ],
    "Resource": [
        "arn:aws:s3:::bucket name",
        "arn:aws:s3:::bucket name/*"
    ]
}
```

To allow any Amazon S3 bucket

If you will use many different buckets to work with your bulk operations metadata, this policy provides access to any bucket:

```
{
    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetBucketLocation",
        "s3:ListBucket",
        "s3:AbortMultipartUpload",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts"
    ],
    "Resource": "*"
}
```

For information about troubleshooting import and export operations, see Troubleshooting bulk import and export.

Running a bulk import job

Bulk import is the action of moving metadata into an AWS IoT SiteWise workspace. For example, bulk import can move metadata from a local file, or a file in an Amazon S3 bucket, to an AWS IoT SiteWise workspace.

Step 1: Prepare the file to import

Download the AWS IoT SiteWise native format file to import assets and the asset models. See AWS IoT SiteWise metadata transfer job schema for more details.

Step 2: Upload the prepared file to Amazon S3

Upload the file to Amazon S3. See Uploading a file to Amazon S3 in the *Amazon Simple Storage Service User Guide* for details.

Import metadata (console)

You can use the AWS IoT SiteWise console to bulk import metadata. Follow Step 1: Prepare the file to import and Step 2: Upload the prepared file to Amazon S3 to prepare a file that is ready to be imported.

Import data from Amazon S3 to AWS IoT SiteWise console

1. Navigate to the AWS IoT SiteWise console.
2. Choose **Bulk operations New** from the navigation pane.
3. Choose **New import** to start the import process.
4. On the **Import metadata** page:
   - Choose **Browse Amazon S3** to view the Amazon S3 bucket and files.
   - Navigate to the Amazon S3 bucket that contains the prepared import file.
   - Select the file to import.
   - Review the selected file, and choose **Import**.
5. The **Bulk operations on SiteWise metadata** page of the AWS IoT SiteWise console displays the newly created import job in the **Jobs progress** table.

Import metadata (AWS CLI)

To perform an import action, use the following procedure:

Import data from Amazon S3 to AWS CLI

1. Create a metadata file that specifies the resources you want to import, following the AWS IoT SiteWise metadata transfer job schema. Store this file in your Amazon S3 bucket.

For examples of metadata files to import, see Import metadata examples.

2. Now create a JSON file with the request body. The request body specifies the source and destination for the transfer job. This file is separate from the file from the previous step. Make sure to specify your Amazon S3 bucket as a source and `iotsitewise` as the destination.

   The following example shows the request body:

```
{
      "metadataTransferJobId": "your-transfer-job-Id",
      "sources": [{
          "type": "s3",
          "s3Configuration": {
              "location": "arn:aws:s3:::your-S3-bucket-name/
your_import_metadata.json"
          }
      }],
      "destination": {
          "type": "iotsitewise"
      }
  }
```

3. Invoke the `CreateMetadataTransferJob` by running the following AWS CLI command. In this example, the request body file from the previous step is named `createMetadataTransferJobExport.json`.

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
  --cli-input-json file://createMetadataTransferJobImport.json
```

   This will create a metadata transfer job, and begin the process of the transferring your selected resources.

Running a bulk export job

Bulk export is the action of moving metadata from an AWS IoT SiteWise workspace to an Amazon S3 bucket.

When you perform a bulk export of your AWS IoT SiteWise content to Amazon S3, you can specify filters to limit which specific asset models and assets you'd like to export.

The filters must be specified in an `iotSiteWiseConfiguration` section within the sources section of your JSON request.

> **Note**
>
> You can include multiple filters in your request. The bulk operation will export asset models and assets that match any of the filters.
> If you don't provide any filters, the bulk operation exports all of your asset models and assets.

Example request body with filters

```
{
      "metadataTransferJobId": "your-transfer-job-id",
      "sources": [
       {
        "type": "iotsitewise",
        "iotSiteWiseConfiguration": {
          "filters": [
           {
              "filterByAssetModel": {
                  "assetModelId": "asset model ID"
              }
            },
            {
              "filterByAssetModel": {
                  "assetModelId": "asset model ID",
                  "includeAssets": true
              }
            },
            {
              "filterByAssetModel": {
                  "assetModelId": "asset model ID",
                  "includeOffspring": true
               }
             }
           ]
          }
        }
       ],
```

```
        "destination": {
           "type": "s3",
           "s3Configuration": {
             "location": "arn:aws:s3:::your-S3-bucket-location"
           }
        }
}
```

Export metadata (console)

The following procedure explains the console export action:

Create an export job in the AWS IoT SiteWise console

1. Navigate to the AWS IoT SiteWise console.
2. Choose **Bulk operations New** from the navigation pane.
3. Choose **New export** to start the export process.
4. On the **Export metadata** page:
   - Enter a name for the export job. This is the name used for the exported file in your Amazon S3 bucket.
   - Choose your resources to export, which sets the filters for the job:
     - Export all assets and asset models. Use filters on assets and asset models.
     - Export assets. Filter on your assets.
       - Select the asset to use for the export filter.
       - (Optional) Add the offspring or the associated asset model.
     - Export asset models. Filter on your asset models.
       - Select the asset model to use for the export filter.
       - (Optional) Add the offspring, or the associated asset or both.
     - Choose **Next**.
   - Navigate to the Amazon S3 bucket:
     - Choose **Browse Amazon S3** to view the Amazon S3 bucket and files.
     - Navigate to the Amazon S3 bucket where the file must be placed.
     - Choose **Next**.
   - Review the export job and choose **Export**.

5. The **Bulk operations on SiteWise metadata** page of the AWS IoT SiteWise console displays the newly created import job in the **Jobs progress** table.

For the different ways to use filters when exporting metadata, see Export metadata examples.

Export metadata (AWS CLI)

The following procedure explains the AWS CLI export action:

Export data from AWS IoT SiteWise to Amazon S3

1. Create a JSON file with your request body. The request body specifies the source and destination for the transfer job. The following example shows an example request body:

```
{
    "metadataTransferJobId": "your-transfer-job-Id",
    "sources": [{
        "type": "iotsitewise"
    }],
    "destination": {
        "type": "s3",
        "s3Configuration": {
            "location": "arn:aws:s3:::your-S3-bucket-location"
        }
    }
}
```

   Make sure to specify your Amazon S3 bucket as the destination of the metadata transfer job.

   > **Note**
   >
   > This example will export all of your asset models and assets. To limit the export to specific asset models or assets, you can include filters in your request body. For more information about applying export filters, see Export metadata examples.

2. Save your request body file to use in the next step. In this example, the file is named `createMetadataTransferJobExport.json`.
3. Invoke the `CreateMetadataTransferJob` by running the following AWS CLI command:

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
        --cli-input-json file://createMetadataTransferJobExport.json
```

Replace the input JSON file `createMetadataTransferJobExport.json` with your own transfer file name.

Jobs progress tracking and error handling

A bulk process job takes time to process. Each job is processed in the order of AWS IoT SiteWise receiving the request. It is processed one-at-a-time for each account. When a job completes, the next in queue automatically starts processing. AWS IoT SiteWise resolves the jobs asynchronously and updates the status of each as it progresses. Each job has a status field that contains the state of the resource and an error message, if applicable.

The state can be one of the following values:

- `VALIDATING` – Validating the job including the submitted file format, and its contents.
- `PENDING` – The job is in a queue. You can cancel jobs in this state from the AWS IoT SiteWise console, but all other states will continue until the end.
- `RUNNING` – Processing the job. It is creating and updating resources as defined by the import file, or exporting resources based on the chosen export job filters. If canceled, any resource imported by this job is not deleted. See Review job progress and details (console) for more information.
- `CANCELLING` – The job is actively being cancelled.
- `ERROR` – One or more resources failed to process. Check the detailed job report for more information. See Inspect error details (console) for more information.
- `COMPLETED` – Job completed without errors.
- `CANCELLED` – The job is cancelled and not queued. If you cancelled a `RUNNING` job, resources already imported by this job at the time of cancellation is not deleted from AWS IoT SiteWise.

Topics

- Jobs progress tracking
- Inspect errors

Jobs progress tracking

Review job progress and details (console)

See Import metadata (console) or Export metadata (console) to start a bulk job.

Job progress overview in the AWS IoT SiteWise console:

1. Navigate to the AWS IoT SiteWise console.
2. Choose **Bulk operations New** from the navigation pane.
3. The **Jobs progress** table in the AWS IoT SiteWise console, displays the list of bulk operation jobs.
4. The **Job type** column describes if it's an export or import job. The **Date imported** columns display the date that the job started.
5. The **Status** column displays the status of the job. You can select a job to see details about the job.
6. The selected job shows **Success** upon being successful, or a list of failure if the job failed. An error description is also displayed with each resource type.

Job details overview in the AWS IoT SiteWise console:

The **Jobs progress** table in the AWS IoT SiteWise console, displays the list of bulk operation jobs.

1. Choose a job to see more details.
2. For an **import** job, the `Data source ARN` represents the Amazon S3 location of the import file.
3. For an **export** job, the `Data destination ARN` represents the Amazon S3 location of the file after the export.
4. The `Status` and `Status reason`, provide additional details on the current job. See Jobs progress tracking and error handling for more details.
5. The `Queued position` represents the position of the job in the process queue. The jobs are processed one at a time. A queued position of 1, indicates that the job will be processed next.
6. The jobs details page also displays the job progress counts.
    - The job progress count types are:
        i. `Total resources` – Indicates the total count of assets in the transfer process.

ii. `Succeeded` – Indicates the count of assets successfully transferred during the process.

iii. `Failed` – Indicates the count assets that failed during the process.

iv. `Skipped` – Indicates the count of assets that were skipped during the process.

7. A job status of `PENDING` or `VALIDATING`, displays all the jobs progress counts as –. This indicates that the jobs progress counts are being evaluated.
8. A job status of `RUNNING` displays the `Total resources` count, the job submitted for processing. The detailed counts (`Succeeded`, `Failed`, and `Skipped`), apply to the processed resources. The sum of the detailed counts is lesser than the `Total resources` count, until the job's status is `COMPLETED` or `ERROR`.
9. If a job's status is `COMPLETED` or `ERROR`, the `Total resources` count equals the sum of the detailed counts (`Succeeded`, `Failed`, and `Skipped`).
10. If a job's status is `ERROR`, check the **Job failures** table for details about the specific errors and failures. See Inspect error details (console) for more details.

Review job progress and details (AWS CLI)

After starting a bulk operation, you can check or update its status using the following API actions:

- To retrieve information on a specific job, use the GetMetadataTransferJob API action.

  Retrieve information with the `GetMetadataTransferJob` API:

  1. Create and run a transfer job. Call the `GetMetadataTransferJob` API.

     Example AWS CLI command:

     ```
     aws iottwinmaker get-metadata-transfer-job \
             --metadata-transfer-job-id your_metadata_transfer_job_id \
             --region your_region
     ```

  2. The `GetMetadataTransferJob` API returns a `MetadataTransferJobProgress` object with the following parameters:
     - **succeededCount** – Indicates the count of assets successfully transferred in the process.
     - **failedCount** – Indicates the count of assets that failed during the process.
     - **skippedCount** – Indicates the count of assets that were skipped during the process.

- **totalCount** – Indicates the total count of assets in the transfer process.

  These parameters indicate the job progress status. If the status is `RUNNING`, they help track the number of resources still to be processed.

  If you encounter schema validation errors, or if **failedCount** is greater than or equal to 1, the job progress state turns to `ERROR`. A full error report for the job is placed in your Amazon S3 bucket. See Inspect errors for more details.

- To list current jobs, use the ListMetadataTransferJobs API action.

  Use a JSON file to filter the returned jobs based on their current state. See the following procedure:

  1. To specify the filters you want to use, create an AWS CLI input JSON file. want to use:

     ```
     {
         "sourceType": "s3",
         "destinationType": "iottwinmaker",
         "filters": [{
             "state": "COMPLETED"
         }]
     }
     ```

     For a list of valid `state` values, see ListMetadataTransferJobsFilter in the *AWS IoT TwinMaker API Reference Guide*.

  2. Use the JSON file as an argument in the following AWS CLI example command:

     ```
     aws iottwinmaker list-metadata-transfer-job --region your_region \
             --cli-input-json file://ListMetadataTransferJobsExample.json
     ```

- To cancel a job, use the CancelMetadataTransferJob API action. This API cancels the specific metadata transfer job, without affecting any resources already exported or imported:

  ```
  aws iottwinmaker cancel-metadata-transfer-job \
          --region your_region \
          --metadata-transfer-job-id job-to-cancel-id
  ```

Inspect errors

Inspect error details (console)

Error details in the AWS IoT SiteWise console:

1. Navigate to the AWS IoT SiteWise console.
2. See the **Jobs progress** table in AWS IoT SiteWise console for a list of bulk operation jobs.
3. Select a job to view the job details.
4. If a job's status is `COMPLETED` or `ERROR`, the `Total resources` count equals the sum of the detailed counts (`Succeeded`, `Failed`, and `Skipped`).
5. If a job's status is `ERROR`, check the **Job failures** table for details about the specific errors and failures.
6. The **Job failures** table displays the content from the job report. The `Resource type` field indicates the location of the error or failures, such as the following:

   - For example, a validation error in the `Bulk operations template` in the `Resource type` field indicates that the import template and metadata schema file format don't match. See AWS IoT SiteWise metadata transfer job schema for more information.
   - A failed `Asset` in the `Resource type` field indicates that the asset is not created because of a conflict with another asset. See Common errors for information on AWS IoT SiteWise resource errors and conflicts.

Inspect error details (AWS CLI)

To handle and diagnose errors produced during a transfer job, see the following procedure about using the `GetMetadataTransferJob` API action:

1. After creating and running a transfer job, call GetMetadataTransferJob:

   ```
   aws iottwinmaker get-metadata-transfer-job \
           --metadata-transfer-job-id your_metadata_transfer_job_id \
           --region us-east-1
   ```

2. Once you see the state of the job turn to `COMPLETED`, you can start verifying the results of the job.
3. When you call `GetMetadataTransferJob`, it returns an object called `MetadataTransferJobProgress`.

The MetadataTransferJobProgress object contains the following parameters:

- **failedCount:** Indicates the count of assets that failed during the transfer process.
- **skippedCount:** Indicates the count of assets that were skipped during the transfer process.
- **succeededCount:** Indicates the count of assets that succeeded during the transfer process.
- **totalCount:** Indicates the total count of assets involved in the transfer process.

4. Additionally, the API call returns an element `reportUrl`, which contains a presigned URL. If your transfer job has any issues that you need to investigate further, visit this url.

Import metadata examples

This section shows how to create metadata files to import asset models and assets with a single bulk import operation.

Example of a bulk import

You can import many asset models and assets with a single bulk import operation. The following example shows how to create a metadata file to do this.

In this example scenario, you have various work sites that contain industrial robots in work cells.

The example defines two asset models:

- `RobotModel1`: This asset model represents a particular type of robot that you have in your work sites. The robot has a measurement property, `Temperature`.
- `WorkCell`: This asset model represents a collection of robots within one of your work sites. The asset model defines a hierarchy, `robotHierarchyOEM1`, to represent the relationship that a work cell contains robots.

The example also defines some assets:

- `WorkCell1`: a work cell within your Boston site
- `RobotArm123456`: a robot within that work cell
- `RobotArm987654`: another robot within that work cell

The following JSON metadata file defines these asset models and assets. Running a bulk import with this metadata creates the asset models and assets within AWS IoT SiteWise, including their hierarchical relationships.

Metadata file for import

```
{
    "assetModels": [
        {
            "assetModelExternalId": "Robot.OEM1.3536",
            "assetModelName": "RobotModel1",
            "assetModelProperties": [
                {
                    "dataType": "DOUBLE",
                    "externalId": "Temperature",
                    "name": "Temperature",
                    "type": {
                        "measurement": {
                            "processingConfig": {
                                "forwardingConfig": {
                                    "state": "ENABLED"
                                }
                            }
                        }
                    },
                    "unit": "fahrenheit"
                }
            ]
        },
        {
            "assetModelExternalId": "ISA95.WorkCell",
            "assetModelName": "WorkCell",
            "assetModelProperties": [],
            "assetModelHierarchies": [
                {
                    "externalId": "workCellHierarchyWithOEM1Robot",
                    "name": "robotHierarchyOEM1",
                    "childAssetModelExternalId": "Robot.OEM1.3536"
                }
            ]
        }
    ],
    "assets": [
```

```
        {
            "assetExternalId": "Robot.OEM1.3536.123456",
            "assetName": "RobotArm123456",
            "assetModelExternalId": "Robot.OEM1.3536"
        },
        {
            "assetExternalId": "Robot.OEM1.3536.987654",
            "assetName": "RobotArm987654",
            "assetModelExternalId": "Robot.OEM1.3536"
        },
        {
            "assetExternalId": "BostonSite.Area1.Line1.WorkCell1",
            "assetName": "WorkCell1",
            "assetModelExternalId": "ISA95.WorkCell",
            "assetHierarchies": [
                {
                    "externalId": "workCellHierarchyWithOEM1Robot",
                    "childAssetExternalId": "Robot.OEM1.3536.123456"
                },
                {
                    "externalId": "workCellHierarchyWithOEM1Robot",
                    "childAssetExternalId": "Robot.OEM1.3536.987654"
                }
            ]
        }
    ]
}
```

Example of initial on-boarding of models and assets

In this example scenario, you have various work sites that contain industrial robots in a company.

The example defines multiple asset models:

- `Sample_Enterprise` – This asset model represents the company that the sites are part of. The asset model defines a hierarchy, `Enterprise to Site`, to represent the relationship of the sites to the enterprise.
- `Sample_Site` – This asset model represents the manufacturing sites within the company. The asset model defines a hierarchy, `Site to Line`, to represent the relationship of the lines to the site.

- `Sample_Welding Line` – This asset model represents an assembly line within work sites. The asset model defines a hierarchy, `Line to Robot`, to represent the relationship of the robots to the line.
- `Sample_Welding Robot` – This asset model represents a particular type of robot in your work sites.

The example also defines assets based on the asset models.

- `Sample_AnyCompany Motor` – This asset is created from `Sample_Enterprise` asset model.
- `Sample_Chicago` – This asset is created from `Sample_Site` asset model.
- `Sample_Welding Line 1` – This asset is created from `Sample_Welding Line` asset model.
- `Sample_Welding Robot 1` – This asset is created from `Sample_Welding Robot` asset model.
- `Sample_Welding Robot 2` – This asset is created from `Sample_Welding Robot` asset model.

The following JSON metadata file defines these asset models and assets. Running a bulk import with this metadata creates the asset models and assets within AWS IoT SiteWise, including their hierarchical relationships.

JSON file to onboard assets and models for import

```
{
    "assetModels": [
        {
            "assetModelExternalId": "External_Id_Welding_Robot",
            "assetModelName": "Sample_Welding Robot",
            "assetModelProperties": [
                {
                    "dataType": "STRING",
                    "externalId": "External_Id_Welding_Robot_Serial_Number",
                    "name": "Serial Number",
                    "type": {
                        "attribute": {
                            "defaultValue": "-"
                        }
                    },
                    "unit": "-"
                },
                {
```

```
                    "dataType": "DOUBLE",
                    "externalId": "External_Id_Welding_Robot_Cycle_Count",
                    "name": "CycleCount",
                    "type": {
                        "measurement": {}
                    },
                    "unit": "EA"
                },
                {
                    "dataType": "DOUBLE",
                    "externalId": "External_Id_Welding_Robot_Joint_1_Current",
                    "name": "Joint 1 Current",
                    "type": {
                        "measurement": {}
                    },
                    "unit": "Amps"
                },
                {
                    "dataType": "DOUBLE",
                    "externalId": "External_Id_Welding_Robot_Joint_1_Max_Current",
                    "name": "Max Joint 1 Current",
                    "type": {
                        "metric": {
                            "expression": "max(joint1current)",
                            "variables": [
                                {
                                    "name": "joint1current",
                                    "value": {
                                        "propertyExternalId":
"External_Id_Welding_Robot_Joint_1_Current"
                                    }
                                }
                            ],
                            "window": {
                                "tumbling": {
                                    "interval": "5m"
                                }
                            }
                        }
                    },
                    "unit": "Amps"
                }
            ]
        },
```

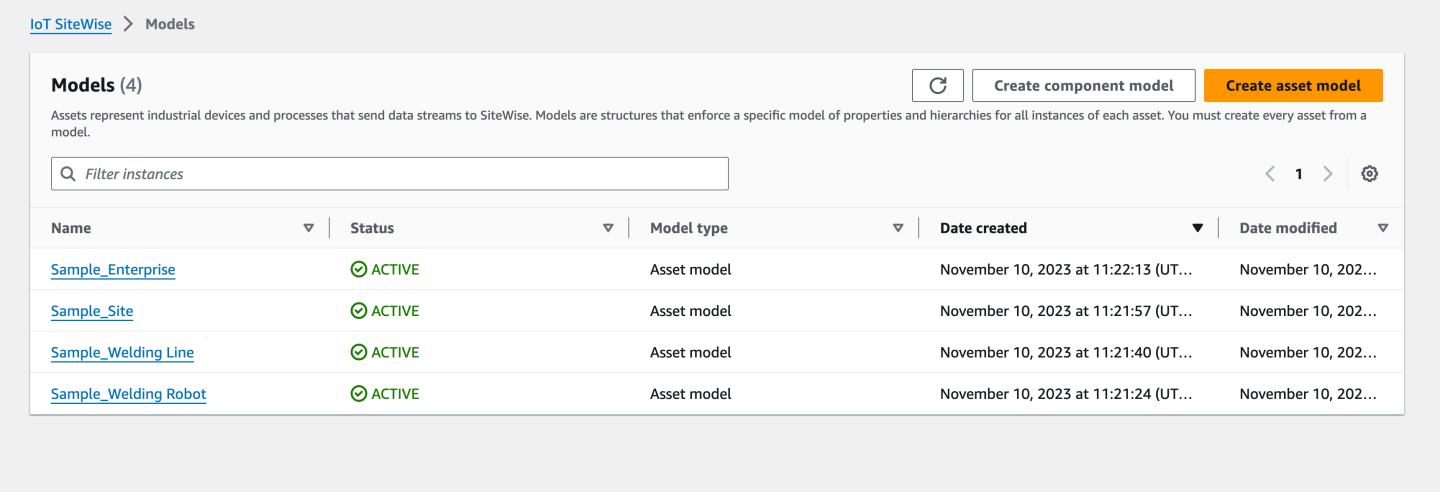
```
        {
            "assetModelExternalId": "External_Id_Welding_Line",
            "assetModelName": "Sample_Welding Line",
            "assetModelProperties": [
                {
                    "dataType": "DOUBLE",
                    "externalId": "External_Id_Welding_Line_Availability",
                    "name": "Availability",
                    "type": {
                        "measurement": {}
                    },
                    "unit": "%"
                }
            ],
            "assetModelHierarchies": [
                {
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "name": "Line to Robot",
                    "childAssetModelExternalId": "External_Id_Welding_Robot"
                }
            ]
        },
        {
            "assetModelExternalId": "External_Id_Site",
            "assetModelName": "Sample_Site",
            "assetModelProperties": [
                {
                    "dataType": "STRING",
                    "externalId": "External_Id_Site_Street_Address",
                    "name": "Street Address",
                    "type": {
                        "attribute": {
                            "defaultValue": "-"
                        }
                    },
                    "unit": "-"
                }
            ],
            "assetModelHierarchies": [
                {
                    "externalId": "External_Id_Site_TO_Line",
                    "name": "Site to Line",
                    "childAssetModelExternalId": "External_Id_Welding_Line"
                }
```

```
            ]
        },
        {
            "assetModelExternalId": "External_Id_Enterprise",
            "assetModelName": "Sample_Enterprise",
            "assetModelProperties": [
                {
                    "dataType": "STRING",
                    "name": "Company Name",
                    "externalId": "External_Id_Enterprise_Company_Name",
                    "type": {
                        "attribute": {
                            "defaultValue": "-"
                        }
                    },
                    "unit": "-"
                }
            ],
            "assetModelHierarchies": [
                {
                    "externalId": "External_Id_Enterprise_TO_Site",
                    "name": "Enterprise to Site",
                    "childAssetModelExternalId": "External_Id_Site"
                }
            ]
        }
    ],
    "assets": [
        {
            "assetExternalId": "External_Id_Welding_Robot_1",
            "assetName": "Sample_Welding Robot 1",
            "assetModelExternalId": "External_Id_Welding_Robot",
            "assetProperties": [
                {
                    "externalId": "External_Id_Welding_Robot_Serial_Number",
                    "attributeValue": "S1000"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Cycle_Count",
                    "alias": "AnyCompany/Chicago/Welding Line/S1000/Count"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Joint_1_Current",
                    "alias": "AnyCompany/Chicago/Welding Line/S1000/1/Current"
```

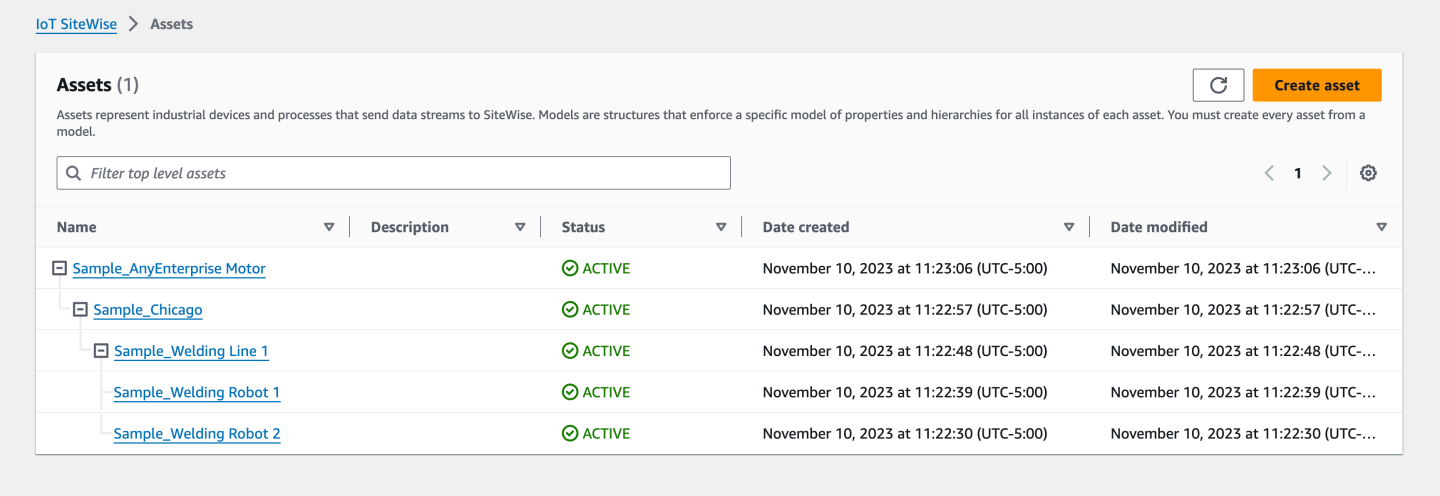
```
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Welding_Robot_2",
            "assetName": "Sample_Welding Robot 2",
            "assetModelExternalId": "External_Id_Welding_Robot",
            "assetProperties": [
                {
                    "externalId": "External_Id_Welding_Robot_Serial_Number",
                    "attributeValue": "S2000"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Cycle_Count",
                    "alias": "AnyCompany/Chicago/Welding Line/S2000/Count"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Joint_1_Current",
                    "alias": "AnyCompany/Chicago/Welding Line/S2000/1/Current"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Welding_Line_1",
            "assetName": "Sample_Welding Line 1",
            "assetModelExternalId": "External_Id_Welding_Line",
            "assetProperties": [
                {
                    "externalId": "External_Id_Welding_Line_Availability",
                    "alias": "AnyCompany/Chicago/Welding Line/Availability"
                }
            ],
            "assetHierarchies": [
                {
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_1"
                },
                {
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_2"
                }
            ]
        },
        {
```

```
            "assetExternalId": "External_Id_Site_Chicago",
            "assetName": "Sample_Chicago",
            "assetModelExternalId": "External_Id_Site",
            "assetHierarchies": [
                {
                    "externalId": "External_Id_Site_TO_Line",
                    "childAssetExternalId": "External_Id_Welding_Line_1"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Enterprise_AnyCompany",
            "assetName": "Sample_AnyEnterprise Motor",
            "assetModelExternalId": "External_Id_Enterprise",
            "assetHierarchies": [
                {
                    "externalId": "External_Id_Enterprise_TO_Site",
                    "childAssetExternalId": "External_Id_Site_Chicago"
                }
            ]
        }
    ]
}
```

The following screenshot is of models that display in the AWS IoT SiteWise console after you run the previous code example.



The following screenshot is of models, assets, and hierarchies that display in the AWS IoT SiteWise console after you run the previous code example.



Example of onboarding additional assets

This example defines additional assets to import to an existing asset model in your account:

- `Sample_Welding Line 2` – This asset is created from `Sample_Welding Line` asset model.
- `Sample_Welding Robot 3`– This asset is created from `Sample_Welding Robot` asset model.
- `Sample_Welding Robot 4`– This asset is created from `Sample_Welding Robot` asset model.

To create the initial assets for this example, see Example of initial on-boarding of models and assets.

The following JSON metadata file defines these asset models and assets. Running a bulk import with this metadata creates the asset models and assets within AWS IoT SiteWise, including their hierarchical relationships.

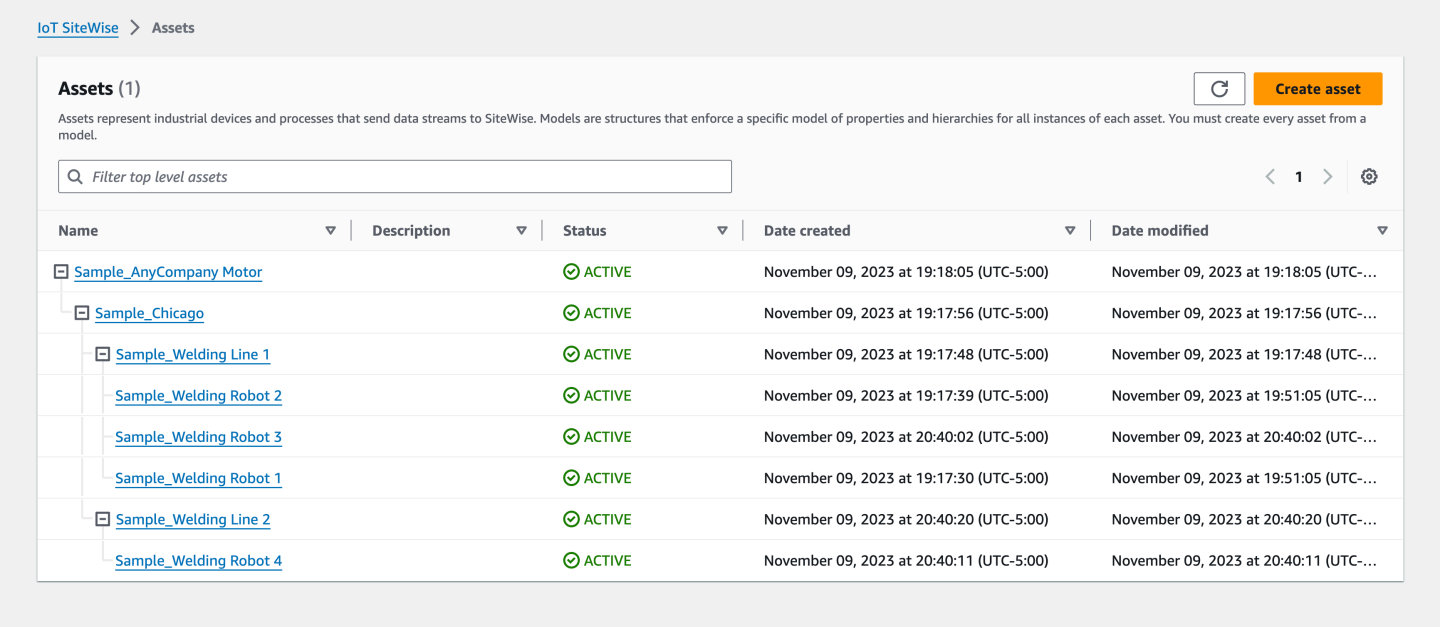
JSON file to onboard additional assets

```
{
    "assets": [
        {
            "assetExternalId": "External_Id_Welding_Robot_3",
            "assetName": "Sample_Welding Robot 3",
            "assetModelExternalId": "External_Id_Welding_Robot",
            "assetProperties": [
                {
                    "externalId": "External_Id_Welding_Robot_Serial_Number",
                    "attributeValue": "S3000"
```

```
                },
                {
                    "externalId": "External_Id_Welding_Robot_Cycle_Count",
                    "alias": "AnyCompany/Chicago/Welding Line/S3000/Count"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Joint_1_Current",
                    "alias": "AnyCompany/Chicago/Welding Line/S3000/1/Current"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Welding_Robot_4",
            "assetName": "Sample_Welding Robot 4",
            "assetModelExternalId": "External_Id_Welding_Robot",
            "assetProperties": [
                {
                    "externalId": "External_Id_Welding_Robot_Serial_Number",
                    "attributeValue": "S4000"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Cycle_Count",
                    "alias": "AnyCompany/Chicago/Welding Line/S4000/Count"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Joint_1_Current",
                    "alias": "AnyCompany/Chicago/Welding Line/S4000/1/Current"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Welding_Line_1",
            "assetName": "Sample_Welding Line 1",
            "assetModelExternalId": "External_Id_Welding_Line",
            "assetHierarchies": [
                {
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_1"
                },
                {
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_2"
                },
                {
```

```
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_3"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Welding_Line_2",
            "assetName": "Sample_Welding Line 2",
            "assetModelExternalId": "External_Id_Welding_Line",
            "assetHierarchies": [
                {
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_4"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Site_Chicago",
            "assetName": "Sample_Chicago",
            "assetModelExternalId": "External_Id_Site",
            "assetHierarchies": [
                {
                    "externalId": "External_Id_Site_TO_Line",
                    "childAssetExternalId": "External_Id_Welding_Line_1"
                },
                {
                    "externalId": "External_Id_Site_TO_Line",
                    "childAssetExternalId": "External_Id_Welding_Line_2"
                }
            ]
        }
    ]
}
```

The following screenshot is of models, assets, and hierarchies that display in the AWS IoT SiteWise console after you run the previous code example.



Example of onboarding new properties

This example defines new properties on existing asset models. See Example of onboarding additional assets to onboard additional assets and models.

- `Joint 1 Temperature` – This property is added to the `Sample_Welding Robot` asset model. This new property will also propagate to each asset created from the `Sample_Welding Robot` asset model.

To add a new property to an existing asset model, see the following JSON metadata file example. As shown in the JSON, the entire existing `Sample_Welding Robot` asset model definition must be provided along with the new property. If the entire property list from the existing definition is not provided, AWS IoT SiteWise deletes the omitted properties.

JSON file to onboard new properties

This example adds a new property `Joint 1 Temperature` to the asset model.

```
{
    "assetModels": [
        {
            "assetModelExternalId": "External_Id_Welding_Robot",
            "assetModelName": "Sample_Welding Robot",
            "assetModelProperties": [
```

```
                {
                    "dataType": "STRING",
                    "externalId": "External_Id_Welding_Robot_Serial_Number",
                    "name": "Serial Number",
                    "type": {
                        "attribute": {
                            "defaultValue": "-"
                        }
                    },
                    "unit": "-"
                },
                {
                    "dataType": "DOUBLE",
                    "externalId": "External_Id_Welding_Robot_Cycle_Count",
                    "name": "CycleCount",
                    "type": {
                        "measurement": {}
                    },
                    "unit": "EA"
                },
                {
                    "dataType": "DOUBLE",
                    "externalId": "External_Id_Welding_Robot_Joint_1_Current",
                    "name": "Joint 1 Current",
                    "type": {
                        "measurement": {}
                    },
                    "unit": "Amps"
                },
                {
                    "dataType": "DOUBLE",
                    "externalId": "External_Id_Welding_Robot_Joint_1_Max_Current",
                    "name": "Max Joint 1 Current",
                    "type": {
                        "metric": {
                            "expression": "max(joint1current)",
                            "variables": [
                                {
                                    "name": "joint1current",
                                    "value": {
                                        "propertyExternalId":
"External_Id_Welding_Robot_Joint_1_Current"
                                    }
                                }
```

```
                        ],
                        "window": {
                            "tumbling": {
                                "interval": "5m"
                            }
                        }
                    }
                },
                "unit": "Amps"
            },
            {
                "dataType": "DOUBLE",
                "externalId": "External_Id_Welding_Robot_Joint_1_Temperature",
                "name": "Joint 1 Temperature",
                "type": {
                    "measurement": {}
                },
                "unit": "degC"
            }
        ]
    }
  ]
}
```

Export metadata examples

When you perform a bulk export of your AWS IoT SiteWise content to Amazon S3, you can specify *filters* to limit which specific asset models and assets you'd like to export.

You specify the filters in an `iotSiteWiseConfiguration` section within the `sources` section of your request body.

> **Note**
>
> You can include multiple filters. The bulk operation will export any asset model or asset that matches any of the filters.
> If you don't provide any filters, then the operation will export all of your asset models and assets.

```
{
```

```
    "metadataTransferJobId": "your-transfer-job-id",
    "sources": [{
        "type": "iotsitewise",
        "iotSiteWiseConfiguration": {
            "filters": [{
                list of filters
            }]
        }
    }],
    "destination": {
        "type": "s3",
        "s3Configuration": {
            "location": "arn:aws:s3:::your-S3-bucket-location"
        }
    }
}
```

Filtering by asset model

You can filter a specific asset model. You can also include all assets using that model, or all asset models within its hierarchy. You can't include both assets and hierarchy.

For more information about hierarchies, see Defining asset model hierarchies.

Asset model

This filter includes the specified asset model:

```
"filterByAssetModel": {
    "assetModelId": "asset model ID"
}
```

Asset model and its assets

This filter includes the specified asset model, along with all assets using that asset model:

```
"filterByAssetModel": {
    "assetModelId": "asset model ID",
    "includeAssets": true
}
```

Asset model and its hierarchy

This filter includes the specified asset model, along with all associated asset models in its hierarchy:

```
"filterByAssetModel": {
    "assetModelId": "asset model ID",
    "includeOffspring": true
}
```

Filtering by asset

You can filter a specific asset. You can also include its asset model, or all associated assets within its hierarchy. You can't include both asset model and hierarchy.

For more information about hierarchies, see Defining asset model hierarchies.

Asset

This filter includes the specified asset:

```
"filterByAsset": {
    "assetId": "asset ID"
}
```

Asset and its asset model

This filter includes the specified asset, along with the asset model it uses:

```
"filterByAsset": {
    "assetId": "asset ID",
    "includeAssetModel": true
}
```

Asset and its hierarchy

This filter includes the specified asset, along with all associated assets in its hierarchy:

```
"filterByAsset": {
    "assetId": "asset ID",
    "includeOffspring": true
```

```
}
```

AWS IoT SiteWise metadata transfer job schema

Use the AWS IoT SiteWise metadata transfer job schema for reference when performing your own bulk import and export operations:

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "IoTSiteWise",
  "description": "Metadata transfer job resource schema for IoTSiteWise",
  "definitions": {
    "Name": {
      "type": "string",
      "minLength": 1,
      "maxLength": 256,
      "pattern": "[^\\u0000-\\u001F\\u007F]+"
    },
    "Description": {
      "type": "string",
      "minLength": 1,
      "maxLength": 2048,
      "pattern": "[^\\u0000-\\u001F\\u007F]+"
    },
    "ID": {
      "type": "string",
      "minLength": 36,
      "maxLength": 36,
      "pattern": "^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$"
    },
    "ExternalId": {
      "type": "string",
      "minLength": 2,
      "maxLength": 128,
      "pattern": "[a-zA-Z0-9_][a-zA-Z_\\-0-9.:]*[a-zA-Z0-9_]+"
    },
    "AttributeValue": {
      "description": "The value of the property attribute.",
      "type": "string",
      "minLength": 1,
      "maxLength": 1024,
      "pattern": "[^\\u0000-\\u001F\\u007F]+"
```

```
    },
    "PropertyUnit": {
      "description": "The unit of measure (such as Newtons or RPM) of the asset
property.",
      "type": "string",
      "minLength": 1,
      "maxLength": 256,
      "pattern": "[^\\u0000-\\u001F\\u007F]+"
    },
    "PropertyAlias": {
      "description": "The property alias that identifies the property.",
      "type": "string",
      "minLength": 1,
      "maxLength": 1000,
      "pattern": "[^\\u0000-\\u001F\\u007F]+"
    },
    "AssetProperty": {
      "description": "The asset property's definition, alias, unit, and notification
state.",
      "type": "object",
      "additionalProperties": false,
      "anyOf": [
        {
          "required": [
            "id"
          ]
        },
        {
          "required": [
            "externalId"
          ]
        }
      ],
      "properties": {
        "id": {
          "description": "The ID of the asset property.",
          "$ref": "#/definitions/ID"
        },
        "externalId": {
          "description": "The ExternalID of the asset property.",
          "$ref": "#/definitions/ExternalId"
        },
        "alias": {
          "$ref": "#/definitions/PropertyAlias"
```

```
        },
        "unit": {
          "$ref": "#/definitions/PropertyUnit"
        },
        "attributeValue": {
          "$ref": "#/definitions/AttributeValue"
        },
        "retainDataOnAliasChange": {
          "type": "string",
          "default": "TRUE",
          "enum": [
            "TRUE",
            "FALSE"
          ]
        },
        "propertyNotificationState": {
          "description": "The MQTT notification state (ENABLED or DISABLED) for this
asset property.",
          "type": "string",
          "enum": [
            "ENABLED",
            "DISABLED"
          ]
        }
      }
    },
    "AssetHierarchy": {
      "description": "A hierarchy specifies allowed parent/child asset relationships.",
      "type": "object",
      "additionalProperties": false,
      "anyOf": [
        {
          "required": [
            "id",
            "childAssetId"
          ]
        },
        {
          "required": [
            "externalId",
            "childAssetId"
          ]
        },
        {
```

```
          "required": [
            "id",
            "childAssetExternalId"
          ]
        },
        {
          "required": [
            "externalId",
            "childAssetExternalId"
          ]
        }
      ],
      "properties": {
        "id": {
          "description": "The ID of a hierarchy in the parent asset's model.",
          "$ref": "#/definitions/ID"
        },
        "externalId": {
          "description": "The ExternalID of a hierarchy in the parent asset's model.",
          "$ref": "#/definitions/ExternalId"
        },
        "childAssetId": {
          "description": "The ID of the child asset to be associated.",
          "$ref": "#/definitions/ID"
        },
        "childAssetExternalId": {
          "description": "The ExternalID of the child asset to be associated.",
          "$ref": "#/definitions/ExternalId"
        }
      }
    },
    "Tag": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "key",
        "value"
      ],
      "properties": {
        "key": {
          "type": "string"
        },
        "value": {
          "type": "string"
```

```
        }
      }
    },
    "AssetModelType": {
      "type": "string",
      "default": null,
      "enum": [
        "ASSET_MODEL",
        "COMPONENT_MODEL"
      ]
    },
    "AssetModelCompositeModel": {
      "description": "Contains a composite model definition in an asset model. This
composite model definition is applied to all assets created from the asset model.",
      "type": "object",
      "additionalProperties": false,
      "anyOf": [
        {
          "required": [
            "id"
          ]
        },
        {
          "required": [
            "externalId"
          ]
        }
      ],
      "required": [
        "name",
        "type"
      ],
      "properties": {
        "id": {
          "description": "The ID of the asset model composite model.",
          "$ref": "#/definitions/ID"
        },
        "externalId": {
          "description": "The ExternalID of the asset model composite model.",
          "$ref": "#/definitions/ExternalId"
        },
        "parentId": {
          "description": "The ID of the parent asset model composite model.",
          "$ref": "#/definitions/ID"
```

```
        },
        "parentExternalId": {
          "description": "The ExternalID of the parent asset model composite model.",
          "$ref": "#/definitions/ExternalId"
        },
        "composedAssetModelId": {
          "description": "The ID of the composed asset model.",
          "$ref": "#/definitions/ID"
        },
        "composedAssetModelExternalId": {
          "description": "The ExternalID of the composed asset model.",
          "$ref": "#/definitions/ExternalId"
        },
        "description": {
          "description": "A description for the asset composite model.",
          "$ref": "#/definitions/Description"
        },
        "name": {
          "description": "A unique, friendly name for the asset composite model.",
          "$ref": "#/definitions/Name"
        },
        "type": {
          "description": "The type of the composite model. For alarm composite models,
this type is AWS/ALARM.",
          "$ref": "#/definitions/Name"
        },
        "properties": {
          "description": "The property definitions of the asset model.",
          "type": "array",
          "items": {
            "$ref": "#/definitions/AssetModelProperty"
          }
        }
      }
    },
    "AssetModelProperty": {
      "description": "Contains information about an asset model property.",
      "type": "object",
      "additionalProperties": false,
      "anyOf": [
        {
          "required": [
            "id"
          ]
```

```
        },
        {
          "required": [
            "externalId"
          ]
        }
      ],
      "required": [
        "name",
        "dataType",
        "type"
      ],
      "properties": {
        "id": {
          "description": "The ID of the asset model property.",
          "$ref": "#/definitions/ID"
        },
        "externalId": {
          "description": "The ExternalID of the asset model property.",
          "$ref": "#/definitions/ExternalId"
        },
        "name": {
          "description": "The name of the asset model property.",
          "$ref": "#/definitions/Name"
        },
        "dataType": {
          "description": "The data type of the asset model property.",
          "$ref": "#/definitions/DataType"
        },
        "dataTypeSpec": {
          "description": "The data type of the structure for this property.",
          "$ref": "#/definitions/Name"
        },
        "unit": {
          "description": "The unit of the asset model property, such as Newtons or
RPM.",
          "type": "string",
          "minLength": 1,
          "maxLength": 256,
          "pattern": "[^\\u0000-\\u001F\\u007F]+"
        },
        "type": {
          "description": "The property type",
          "$ref": "#/definitions/PropertyType"
```

```
        }
      }
    },
    "DataType": {
      "type": "string",
      "enum": [
        "STRING",
        "INTEGER",
        "DOUBLE",
        "BOOLEAN",
        "STRUCT"
      ]
    },
    "PropertyType": {
      "description": "Contains a property type, which can be one of attribute,
measurement, metric, or transform.",
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "attribute": {
          "$ref": "#/definitions/Attribute"
        },
        "transform": {
          "$ref": "#/definitions/Transform"
        },
        "metric": {
          "$ref": "#/definitions/Metric"
        },
        "measurement": {
          "$ref": "#/definitions/Measurement"
        }
      }
    },
    "Attribute": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "defaultValue": {
          "type": "string",
          "minLength": 1,
          "maxLength": 1024,
          "pattern": "[^\\u0000-\\u001F\\u007F]+"
        }
      }
```

```
    },
    "Transform": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "expression",
        "variables"
      ],
      "properties": {
        "expression": {
          "description": "The mathematical expression that defines the transformation
function.",
          "type": "string",
          "minLength": 1,
          "maxLength": 1024
        },
        "variables": {
          "description": "The list of variables used in the expression.",
          "type": "array",
          "items": {
            "$ref": "#/definitions/ExpressionVariable"
          }
        },
        "processingConfig": {
          "$ref": "#/definitions/TransformProcessingConfig"
        }
      }
    },
    "TransformProcessingConfig": {
      "description": "The processing configuration for the given transform property.",
      "type": "object",
      "additionalProperties": false,
      "required": [
        "computeLocation"
      ],
      "properties": {
        "computeLocation": {
          "description": "The compute location for the given transform property.",
          "$ref": "#/definitions/ComputeLocation"
        },
        "forwardingConfig": {
          "description": "The forwarding configuration for a given property.",
          "$ref": "#/definitions/ForwardingConfig"
        }
```

```
      }
    },
    "Metric": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "expression",
        "variables",
        "window"
      ],
      "properties": {
        "expression": {
          "description": "The mathematical expression that defines the metric
aggregation function.",
          "type": "string",
          "minLength": 1,
          "maxLength": 1024
        },
        "variables": {
          "description": "The list of variables used in the expression.",
          "type": "array",
          "items": {
            "$ref": "#/definitions/ExpressionVariable"
          }
        },
        "window": {
          "description": "The window (time interval) over which AWS IoT SiteWise
computes the metric's aggregation expression",
          "$ref": "#/definitions/MetricWindow"
        },
        "processingConfig": {
          "$ref": "#/definitions/MetricProcessingConfig"
        }
      }
    },
    "MetricProcessingConfig": {
      "description": "The processing configuration for the metric.",
      "type": "object",
      "additionalProperties": false,
      "required": [
        "computeLocation"
      ],
      "properties": {
        "computeLocation": {
```

```
          "description": "The compute location for the given metric property.",
          "$ref": "#/definitions/ComputeLocation"
        }
      }
    },
    "ComputeLocation": {
      "type": "string",
      "enum": [
        "EDGE",
        "CLOUD"
      ]
    },
    "ForwardingConfig": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "state"
      ],
      "properties": {
        "state": {
          "type": "string",
          "enum": [
            "ENABLED",
            "DISABLED"
          ]
        }
      }
    },
    "MetricWindow": {
      "description": "Contains a time interval window used for data aggregate
computations (for example, average, sum, count, and so on).",
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "tumbling": {
          "description": "The tumbling time interval window.",
          "type": "object",
          "additionalProperties": false,
          "required": [
            "interval"
          ],
          "properties": {
            "interval": {
              "description": "The time interval for the tumbling window.",
```

```
              "type": "string",
              "minLength": 2,
              "maxLength": 23
            },
            "offset": {
              "description": "The offset for the tumbling window.",
              "type": "string",
              "minLength": 2,
              "maxLength": 25
            }
          }
        }
      }
    },
    "ExpressionVariable": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "name",
        "value"
      ],
      "properties": {
        "name": {
          "description": "The friendly name of the variable to be used in the
expression.",
          "type": "string",
          "minLength": 1,
          "maxLength": 64,
          "pattern": "^[a-z][a-z0-9_]*$"
        },
        "value": {
          "description": "The variable that identifies an asset property from which to
use values.",
          "$ref": "#/definitions/VariableValue"
        }
      }
    },
    "VariableValue": {
      "type": "object",
      "additionalProperties": false,
      "anyOf": [
        {
          "required": [
            "propertyId"
```

```
          ]
        },
        {
          "required": [
            "propertyExternalId"
          ]
        }
      ],
      "properties": {
        "propertyId": {
          "$ref": "#/definitions/ID"
        },
        "propertyExternalId": {
          "$ref": "#/definitions/ExternalId"
        },
        "hierarchyId": {
          "$ref": "#/definitions/ID"
        },
        "hierarchyExternalId": {
          "$ref": "#/definitions/ExternalId"
        }
      }
    },
    "Measurement": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "processingConfig": {
          "$ref": "#/definitions/MeasurementProcessingConfig"
        }
      }
    },
    "MeasurementProcessingConfig": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "forwardingConfig"
      ],
      "properties": {
        "forwardingConfig": {
          "description": "The forwarding configuration for the given measurement
property.",
          "$ref": "#/definitions/ForwardingConfig"
        }
```

```
      }
    },
    "AssetModelHierarchy": {
      "description": "Contains information about an asset model hierarchy.",
      "type": "object",
      "additionalProperties": false,
      "anyOf": [
        {
          "required": [
            "id",
            "childAssetModelId"
          ]
        },
        {
          "required": [
            "id",
            "childAssetModelExternalId"
          ]
        },
        {
          "required": [
            "externalId",
            "childAssetModelId"
          ]
        },
        {
          "required": [
            "externalId",
            "childAssetModelExternalId"
          ]
        }
      ],
      "required": [
        "name"
      ],
      "properties": {
        "id": {
          "description": "The ID of the asset model hierarchy.",
          "$ref": "#/definitions/ID"
        },
        "externalId": {
          "description": "The ExternalID of the asset model hierarchy.",
          "$ref": "#/definitions/ExternalId"
        },
```

```
        "name": {
          "description": "The name of the asset model hierarchy.",
          "$ref": "#/definitions/Name"
        },
        "childAssetModelId": {
          "description": "The ID of the asset model. All assets in this hierarchy must
 be instances of the child AssetModelId asset model.",
          "$ref": "#/definitions/ID"
        },
        "childAssetModelExternalId": {
          "description": "The ExternalID of the asset model. All assets in this
 hierarchy must be instances of the child AssetModelId asset model.",
          "$ref": "#/definitions/ExternalId"
        }
      }
    },
    "AssetModel": {
      "type": "object",
      "additionalProperties": false,
      "anyOf": [
        {
          "required": [
            "assetModelId"
          ]
        },
        {
          "required": [
            "assetModelExternalId"
          ]
        }
      ],
      "required": [
        "assetModelName"
      ],
      "properties": {
        "assetModelId": {
          "description": "The ID of the asset model.",
          "$ref": "#/definitions/ID"
        },
        "assetModelExternalId": {
          "description": "The ID of the asset model.",
          "$ref": "#/definitions/ExternalId"
        },
        "assetModelName": {
```

```
          "description": "A unique, friendly name for the asset model.",
          "$ref": "#/definitions/Name"
        },
        "assetModelDescription": {
          "description": "A description for the asset model.",
          "$ref": "#/definitions/Description"
        },
        "assetModelType": {
          "description": "The type of the asset model.",
          "$ref": "#/definitions/AssetModelType"
        },
        "assetModelProperties": {
          "description": "The property definitions of the asset model.",
          "type": "array",
          "items": {
            "$ref": "#/definitions/AssetModelProperty"
          }
        },
        "assetModelCompositeModels": {
          "description": "The composite asset models that are part of this asset model.
Composite asset models are asset models that contain specific properties.",
          "type": "array",
          "items": {
            "$ref": "#/definitions/AssetModelCompositeModel"
          }
        },
        "assetModelHierarchies": {
          "description": "The hierarchy definitions of the asset model. Each hierarchy
specifies an asset model whose assets can be children of any other assets created from
this asset model.",
          "type": "array",
          "items": {
            "$ref": "#/definitions/AssetModelHierarchy"
          }
        },
        "tags": {
          "description": "A list of key-value pairs that contain metadata for the asset
model.",
          "type": "array",
          "items": {
            "$ref": "#/definitions/Tag"
          }
        }
      }
```

```
    },
    "Asset": {
      "type": "object",
      "additionalProperties": false,
      "anyOf": [
        {
          "required": [
            "assetId",
            "assetModelId"
          ]
        },
        {
          "required": [
            "assetExternalId",
            "assetModelId"
          ]
        },
        {
          "required": [
            "assetId",
            "assetModelExternalId"
          ]
        },
        {
          "required": [
            "assetExternalId",
            "assetModelExternalId"
          ]
        }
      ],
      "required": [
        "assetName"
      ],
      "properties": {
        "assetId": {
          "description": "The ID of the asset",
          "$ref": "#/definitions/ID"
        },
        "assetExternalId": {
          "description": "The external ID of the asset",
          "$ref": "#/definitions/ExternalId"
        },
        "assetModelId": {
          "description": "The ID of the asset model from which to create the asset.",
```

```
          "$ref": "#/definitions/ID"
        },
        "assetModelExternalId": {
          "description": "The ExternalID of the asset model from which to create the
asset.",
          "$ref": "#/definitions/ExternalId"
        },
        "assetName": {
          "description": "A unique, friendly name for the asset.",
          "$ref": "#/definitions/Name"
        },
        "assetDescription": {
          "description": "A description for the asset",
          "$ref": "#/definitions/Description"
        },
        "assetProperties": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/AssetProperty"
          }
        },
        "assetHierarchies": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/AssetHierarchy"
          }
        },
        "tags": {
          "description": "A list of key-value pairs that contain metadata for the
asset.",
          "type": "array",
          "uniqueItems": false,
          "items": {
            "$ref": "#/definitions/Tag"
          }
        }
      }
    }
  },
  "additionalProperties": false,
  "properties": {
    "assetModels": {
      "type": "array",
      "uniqueItems": false,
```

```
      "items": {
        "$ref": "#/definitions/AssetModel"
      }
    },
    "assets": {
      "type": "array",
      "uniqueItems": false,
      "items": {
        "$ref": "#/definitions/Asset"
      }
    }
  }
}
```

Monitoring data with alarms

You can configure alarms for your data to alert your team when equipment or processes perform sub-optimally. Optimal performance of a machine or process means that the values for certain metrics should be within a range of high and low limits. When these metrics are outside their operating range, equipment operators must be notified so they can fix the issue. Use alarms to quickly identify issues and notify operators to maximize performance of your equipment and processes.

Topics

Alarm types

You can define alarms that detect in the AWS Cloud and alarms that you detect with external processes. AWS IoT SiteWise supports the following types of alarms:

- **AWS IoT Events alarms**

  AWS IoT Events alarms are alarms that detect in AWS IoT Events. AWS IoT SiteWise sends asset property values to an alarm model in AWS IoT Events. Then, AWS IoT Events sends the alarm state to AWS IoT SiteWise. You can configure options such as when the alarm detects and whom to notify when the alarm state changes. You can also define the AWS IoT Events actions that occur when the alarm state changes.

  Alarms in AWS IoT Events are instances of alarm models. The alarm model specifies the threshold and severity of the alarm, what to do when the alarm state changes, and more. When you configure each trait of the alarm model, you specify an attribute property from the asset model that the alarm monitors. All assets based on the asset model use the value of the

attribute when AWS IoT Events evaluates that trait of the alarm. For more information, see Using alarms in the *AWS IoT Events Developer Guide*.

You can respond to an AWS IoT Events alarm when it changes state. For example, you can acknowledge or snooze an alarm when it becomes active. You can also enable, disable, and reset alarms.

SiteWise Monitor users can visualize, configure, and respond to AWS IoT Events alarms in SiteWise Monitor portals. For more information, see Monitoring with alarms in the *AWS IoT SiteWise Monitor Application Guide*.

> **Note**
>
> AWS IoT Events charges apply to evaluate these alarms and transfer data between AWS IoT SiteWise and AWS IoT Events. For more information, see AWS IoT Events pricing.

- **External alarms**

  External alarms are alarms that you evaluate outside of AWS IoT SiteWise. Use external alarms if you have a data source that reports alarm state. The external alarm contains a measurement property to which you ingest the alarm state data.

  You can't acknowledge or snooze an external alarm when it changes state.

  SiteWise Monitor users can see the state of external alarms in SiteWise Monitor portals, but they can't configure or respond to these alarms.

  AWS IoT SiteWise doesn't evaluate the state of external alarms.

Alarm states

Industrial alarms include information about the state of the equipment or process they monitor and (optional) information about the operator's response to the alarm state.

When you define an AWS IoT Events alarm, you specify whether or not to enable the *acknowledge flow*. The acknowledge flow is enabled by default. When you enable this option, operators can acknowledge the alarm and leave a note with details about the alarm or the actions they took to address it. If an operator doesn't acknowledge an active alarm before it becomes inactive, the alarm becomes latched. The latched state indicates that the alarm became active and wasn't

acknowledged, so an operator needs to check on the equipment or process and acknowledge the latched alarm.

Alarms have the following states:

- **Normal** (`Normal`) – The alarm is enabled but inactive. The industrial process or equipment operates as expected.
- **Active** (`Active`) – The alarm is active. The industrial process or equipment is outside its operating range and needs attention.
- **Acknowledged** (`Acknowledged`) – An operator acknowledged the state of the alarm.

  This state applies to only alarms where you enable the acknowledge flow.
- **Latched** (`Latched`) – The alarm returned to normal but was active and no operator acknowledged it. The industrial process or equipment requires attention from an operator to reset the alarm to normal.

  This state applies to only alarms where you enable the acknowledge flow.
- **Snoozed** (`SnoozeDisabled`) – The alarm is disabled because an operator snoozed the alarm. The operator defines the duration for which the alarm snoozes. After that duration, the alarm returns to normal state.
- **Disabled** (`Disabled`) – The alarm is disabled and won't detect.

Alarm state properties

AWS IoT SiteWise stores alarm state data as a JSON object serialized to a string. This object contains the state and additional information about the alarm, such as operator response actions and the rule that the alarm evaluates.

You identify the alarm state property by its name and structure type, `AWS/ALARM_STATE`. For more information, see Defining alarms on asset models.

The alarm state data object contains the following information:

`stateName`

The state of the alarm. For more information, see Alarm states.

Data type: `STRING`

`customerAction`

(Optional) An object that contains information about an operator's response to the alarm. Operators can enable, disable, acknowledge, and snooze alarms. When they do so, the alarm state data includes their response and the note that they can leave when they respond. This object contains the following information:

`actionName`

The name of the action that the operator takes to respond to the alarm. This value contains one of the following strings:

- `ENABLE`
- `DISABLE`
- `SNOOZE`
- `ACKNOWLEDGE`
- `RESET`

Data type: `STRING`

`enable`

(Optional) An object that is present in `customerAction` when the operator enables the alarm. When an operator enables the alarm, the alarm state changes to `Normal`. This object contains the following information:

`note`

(Optional) The note that the customer leaves when they enable the alarm.

Data type: `STRING`

Maximum length: 128 characters

`disable`

(Optional) An object that is present in `customerAction` when the operator disables the alarm. When an operator enables the alarm, the alarm state changes to `Disabled`. This object contains the following information:

`note`

(Optional) The note that the customer leaves when they disable the alarm.

Data type: `STRING`

Maximum length: 128 characters

`acknowledge`

(Optional) An object that is present in `customerAction` when the operator acknowledges the alarm. When an operator enables the alarm, the alarm state changes to `Acknowledged`. This object contains the following information:

`note`

(Optional) The note that the customer leaves when they acknowledge the alarm.

Data type: `STRING`

Maximum length: 128 characters

`snooze`

(Optional) An object that is present in `customerAction` when the operator snoozes the alarm. When an operator enables the alarm, the alarm state changes to `SnoozeDisabled`. This object contains the following information:

`snoozeDuration`

The duration in seconds that the operator snoozes the alarm. The alarm changes to `Normal` state after this duration.

Data type: `INTEGER`

`note`

(Optional) The note that the customer leaves when they snooze the alarm.

Data type: `STRING`

Maximum length: 128 characters

`ruleEvaluation`

(Optional) An object that contains information about the rule that evaluates the alarm. This object contains the following information:

`simpleRule`

An object that contains information about a simple rule, which compares a property value to a threshold value with a comparison operator. This object contains the following information:

`inputProperty`

The value of the property that this alarm evaluates.

Data type: `DOUBLE`

`operator`

The comparison operator that this alarm uses to compare the property with the threshold. This value contains one of the following strings:

- < – Less than
- <= – Less than or equal
- == – Equal
- != – Not equal
- >= – Greater than or equal
- > – Greater than

Data type: `STRING`

`threshold`

The threshold value that this alarm compares the property value against.

Data type: `DOUBLE`

Defining alarms on asset models

Asset models drive standardization of your industrial data and alarms. You can define alarm definitions on asset models to standardize the alarms for all assets based on an asset model.

You use *composite asset models* to define alarms on asset models. Composite asset models are asset models that standardize a specific set of properties on another asset model. Composite asset models ensure that certain properties are present on an asset model. Alarms have type, state, and (optional) source properties, so the alarm composite model enforces that these properties exist.

Each composite asset model has a type that defines the properties for that composite model. Alarm composite models define properties for alarm type, alarm state, and (optional) alarm source. When you create an asset from an asset model with composite models, the asset includes the properties from the composite model alongside the properties that you specify in the asset model.

Each property in a composite model must have the name that identifies it for its type of composite model. Composite model properties support properties with complex data types. These properties have the `STRUCT` data type and a `dataTypeSpec` trait that specifies the complex data type of the property. Complex data type properties contain JSON data serialized as strings.

Alarm composite models have the following properties. Each property must have the name that identifies it for this type of composite model.

Alarm type

The type of the alarm. Specify one of the following:

- `IOT_EVENTS` – An AWS IoT Events alarm. AWS IoT SiteWise sends data to AWS IoT Events to evaluate the state of this alarm. You must specify the alarm source property to define the AWS IoT Events alarm model for this alarm definition.
- `EXTERNAL` – An external alarm. You ingest the state of the alarm as a measurement.

Property name: `AWS/ALARM_TYPE`

Property type: attribute

Data type: `STRING`

Alarm state

The time series data for the state of the alarm. This is an object serialized as a string that contains the state and other information about the alarm. For more information, see Alarm state properties.

Property name: `AWS/ALARM_STATE`

Property type: measurement

Data type: `STRUCT`

Data structure type: `AWS/ALARM_STATE`

Alarm source

(Optional) The Amazon Resource Name (ARN) of the resource that evaluates the state of the alarm. For AWS IoT Events alarms, this is the ARN of the alarm model.

Property name: `AWS/ALARM_SOURCE`

Property type: attribute

Data type: `STRING`

Example Example alarm composite model

The following asset model represents a boiler that has an alarm to monitor its temperature. AWS IoT SiteWise sends the temperature data to AWS IoT Events to detect the alarm.

```
{
  "assetModelName": "Boiler",
  "assetModelDescription": "A boiler that alarms when its temperature exceeds its
 limit.",
  "assetModelProperties": [
    {
      "name": "Temperature",
      "dataType": "DOUBLE",
      "unit": "Celsius",
      "type": {
        "measurement": {}
      }
    },
    {
      "name": "High Temperature",
      "dataType": "DOUBLE",
      "unit": "Celsius",
      "type": {
        "attribute": {
          "defaultValue": "105.0"
        }
      }
    }
  ],
  "assetModelCompositeModels": [
    {
      "name": "BoilerTemperatureHighAlarm",
      "type": "AWS/ALARM",
      "properties": [
        {
          "name": "AWS/ALARM_TYPE",
          "dataType": "STRING",
          "type": {
            "attribute": {
              "defaultValue": "IOT_EVENTS"
            }
```

```
          }
        },
        {
          "name": "AWS/ALARM_STATE",
          "dataType": "STRUCT",
          "dataTypeSpec": "AWS/ALARM_STATE",
          "type": {
            "measurement": {}
          }
        },
        {
          "name": "AWS/ALARM_SOURCE",
          "dataType": "STRING",
          "type": {
            "attribute": {}
          }
        }
      ]
    }
  ]
}
```

Topics

- Defining AWS IoT Events alarms
- Defining external alarms

Defining AWS IoT Events alarms

When you create an AWS IoT Events alarm, AWS IoT SiteWise sends asset property values to AWS IoT Events to evaluate the state of the alarm. AWS IoT Events alarm definitions depend on an alarm model that you define in AWS IoT Events. To define an AWS IoT Events alarm on an asset model, you define an alarm composite model that specifies the AWS IoT Events alarm model as its alarm source property.

AWS IoT Events alarms depend on inputs such as alarm thresholds and alarm notification settings. You define these inputs as attributes on the asset model. You can then customize these inputs on each asset based on the model. The AWS IoT SiteWise console can create these attributes for you. If you define alarms with the AWS CLI or API, you must manually define these attributes on the asset model.

You can also define other actions that happen when your alarm detects, such as custom alarm notification actions. For example, you can configure an action that sends a push notification to an Amazon SNS topic. For more information the actions that you can define, see Working with other AWS services in the *AWS IoT Events Developer Guide*.

When you update or delete an asset model, AWS IoT SiteWise can check if an alarm model in AWS IoT Events is monitoring an asset property associated with this asset model. This prevents you from deleting an asset property that an AWS IoT Events alarm is currently using. To enable this feature in AWS IoT SiteWise, you must have the `iotevents:ListInputRoutings` permission. This permission allows AWS IoT SiteWise to make calls to the ListInputRoutings API operation supported by AWS IoT Events. For more information, see (Optional) ListInputRoutings permission.

> **Note**
>
> The alarm notifications feature isn't available in the China (Beijing) Region.

Topics

- Requirements for alarm notifications
- Defining an AWS IoT Events alarm (AWS IoT SiteWise console)
- Defining an AWS IoT Events alarm (AWS IoT Events console)
- Defining an AWS IoT Events alarm (AWS CLI)

Requirements for alarm notifications

AWS IoT Events uses an AWS Lambda function in your AWS account to send alarm notifications. You must create this Lambda function in the same AWS Region as your alarms to enable alarm notifications. This Lambda function uses Amazon Simple Notification Service (Amazon SNS) to send text notifications and Amazon Simple Email Service (Amazon SES) to send email notifications. When you create the AWS IoT Events alarm, you configure the protocols and settings that the alarm uses to send notifications.

AWS IoT Events provides an AWS CloudFormation stack template that you can use to create this Lambda function in your account. For more information, see Alarm notification Lambda function in the *AWS IoT Events Developer Guide*.

Defining an AWS IoT Events alarm (AWS IoT SiteWise console)

You can use the AWS IoT SiteWise console to define an AWS IoT Events alarm on an existing asset model. To define an AWS IoT Events alarm on a new asset model, create the asset model, and then complete these steps. For more information, see Creating asset models.

> **Important**
>
> Each alarm requires an attribute that specifies the threshold value to compare against for the alarm. You must define the threshold value attribute on the asset model before you can define an alarm.
>
> Consider an example where you want to define an alarm that detects when a wind turbine exceeds its maximum wind speed rating of 50 mph. Before you define the alarm, you must define an attribute (**Maximum wind speed**) with a default value of `50`.

To define an AWS IoT Events alarm on an asset model

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Models**.
3. Choose the asset model for which to define an alarm.
4. Choose the **Alarm** tab.
5. Choose **Add alarm**.
6. In the **Alarm type options** section, choose **AWS IoT Events alarm**.
7. In the **Alarm details** section, do the following:
   a. Enter a name for your alarm.
   b. (Optional) Enter a description for your alarm.
8. In the **Threshold definitions** section, you define when the alarm detects and the severity of the alarm. Do the following:
   a. Select the **Property** on which the alarm detects. Each time this property receives a new value, AWS IoT SiteWise sends the value to AWS IoT Events to evaluate the state of the alarm.
   b. Select the **Operator** to use to compare the property with the threshold value. Choose from the following options:

- **< less than**
- **<= less than or equal**
- **== equal**
- **!= not equal**
- **>= greater than or equal**
- **> greater than**

c. For **Value**, select the attribute property to use as the threshold value. AWS IoT Events compares the value of the property with the value of this attribute.

d. Enter the **Severity** of the alarm. Use a number that your team understands to reflect the severity of this alarm.

9. (Optional) In the **Notification settings - *optional*** section, do the following:

a. Choose **Active**.

> **Note**
>
> If you choose **Inactive**, you and your team won't receive any alarm notifications.

b. For **Recipient**, choose the recipient.

> **Important**
>
> You can send alarm notifications to AWS IAM Identity Center users. To use this feature, you must enable IAM Identity Center. You can only enable IAM Identity Center in one AWS Region at a time. This means that you can define alarm notifications only in the Region where you enable IAM Identity Center. For more information, see Getting started in the *AWS IAM Identity Center User Guide*.

c. For **Protocol**, choose from the following options:

- **Email & text** – The alarm notifies IAM Identity Center users with an SMS message and an email message.
- **Email** – The alarm notifies IAM Identity Center users with an email message.
- **Text** – The alarm notifies IAM Identity Center users with an SMS message.

d. For **Sender**, choose the sender.

> **Important**
>
> You must verify the sender email address in Amazon Simple Email Service (Amazon SES). For more information, see Verifying email addresses in Amazon SES, in the *Amazon Simple Email Service Developer Guide*.

10. In the **Default asset state** section, you can set the default state for alarms created from this asset model.

> **Note**
>
> You activate or deactivate this alarm for assets that you create from this asset model in a later step.

11. In the **Advanced settings** section, you can configure the permissions, the additional notification settings, the alarm state actions, the alarm model in SiteWise Monitor, and the acknowledge flow.

> **Note**
>
> AWS IoT Events alarms require the following service roles:
>
> - A role that AWS IoT Events assumes to send alarm state values to AWS IoT SiteWise.
> - A role that AWS IoT Events assumes to send data to Lambda. You only need this role if your alarm sends notifications.

In the **Permissions** section, do the following:

a. For **AWS IoT Events role**, use an existing role or create a role with the required permissions. This role requires the `iotsitewise:BatchPutAssetPropertyValue` permission and a trust relationship that allows iotevents.amazonaws.com to assume the role.

b. For the **AWS IoT Events Lambda role**, use an existing role or create a role with the required permissions. This role requires the `lambda:InvokeFunction` and `sso-directory:DescribeUser` permissions and a trust relationship that allows `iotevents.amazonaws.com` to assume the role.

12. (Optional) In the **Additional notification settings** section, do the following:

    a. For **Recipient attribute**, you define an attribute whose value specifies the recipient of the notification. You can choose IAM Identity Center users as recipients.

       You can create an attribute or use an existing attribute on the asset model.

       - If you choose **Create a new recipient attribute**, specify the **Recipient attribute name** and **Recipient default value - optional** for the attribute.
       - If you choose **Use an existing recipient attribute**, choose the attribute in **Recipient attribute name**. The alarm uses the default value of the attribute that you choose.

       You can override the default value on each asset that you create from this asset model.

    b. For **Custom message attribute**, you define an attribute whose value specifies the custom message to send in addition to the default state change message. For example, you can specify a message that helps your team understand how to address this alarm.

       You can choose to create an attribute or use an existing attribute on the asset model.

       - If you choose to **Create a new custom message attribute**, specify the **Custom message attribute name** and **Custom message default value - *optional*** for the attribute.
       - If you choose **Use an existing custom message attribute**, choose the attribute in **Custom message attribute name**. The alarm uses the default value of the attribute that you choose.

       You can override the default value on each asset that you create from this asset model.

    c. For **Manage your Lambda function**, do one of the following:

       - To have AWS IoT SiteWise create a new Lambda function, choose **Create a new lambda from an AWS managed template**.
       - To use an existing Lambda function, choose **Use an existing lambda** and choose the name of the function.

       For more information, see Managing alarm notifications in the *AWS IoT Events Developer Guide*.

13. (Optional) In the **Set state action** section, do the following:

a. Choose **Edit action**.

b. Under **Add alarm state actions**, add actions. and the choose **Save**.

You can add up to 10 actions.

AWS IoT Events can perform actions when the alarm is active. You can define built-in actions to use a timer or set a variable, or send data to other AWS resources. For more information, see Supported actions in the *AWS IoT Events Developer Guide*.

14. (Optional) Under **Manage alarm model in SiteWise Monitor - *optional***, choose **Active** or **Inactive**.

    Use this option so that you can update the alarm model in SiteWise Monitorss. This option is enabled by default.

15. Under **Acknowledge flow**, choose **Active** or **Inactive**. For more information about the acknowledge flow, see Alarm states.

16. Choose **Add alarm**.

> **Note**
>
> The AWS IoT SiteWise console makes multiple API requests to add the alarm to the asset model. When you choose **Add alarm**, the console opens a dialog box that shows the progress of these API requests. Stay on this page until each API requests succeeds or until an API request fails. If a request fails, close the dialog box, fix the issue, and choose **Add alarm** to try again.

Defining an AWS IoT Events alarm (AWS IoT Events console)

You can use the AWS IoT Events console to define an AWS IoT Events alarm on an existing asset model. To define an AWS IoT Events alarm on a new asset model, create the asset model, and then complete these steps. For more information, see Creating asset models.

> **Important**
>
> Each alarm requires an attribute that specifies the threshold value to compare against for the alarm. You must define the threshold value attribute on the asset model before you can define an alarm.
> Consider an example where you want to define an alarm that detects when a wind turbine exceeds its maximum wind speed rating of 50 mph. Before you define the alarm, you must define an attribute (**Maximum wind speed**) with a default value of `50`.

To define an AWS IoT Events alarm on an asset model

1. Navigate to the AWS IoT Events console.
2. In the navigation pane, choose **Alarm models**.
3. Choose **Create alarm model**.
4. Enter a name for your alarm.
5. (Optional) Enter a description for your alarm.
6. In the **Alarm target** section, do the following:
   a. For **Target options**, choose **AWS IoT SiteWise asset property**.
   b. Choose the asset model for which you want to add the alarm.
7. In the **Threshold definitions** section, you define when the alarm detects and the severity of the alarm. Do the following:
   a. Select the **Property** on which the alarm detects. Each time this property receives a new value, AWS IoT SiteWise sends the value to AWS IoT Events to evaluate the state of the alarm.
   b. Select the **Operator** to use to compare the property with the threshold value. Choose from the following options:
      - **< less than**
      - **<= less than or equal**
      - **== equal**
      - **!= not equal**
      - **>= greater than or equal**

- **> greater than**

c. For **Value**, select the attribute property to use as the threshold value. AWS IoT Events compares the value of the property with the value of this attribute.

d. Enter the **Severity** of the alarm. Use a number that your team understands to reflect the severity of this alarm.

8. (Optional) In the **Notification settings - *optional*** section, do the following:

a. For **Protocol**, choose from the following options:

- **Email & text** – The alarm notifies IAM Identity Center users with an SMS message and an email message.
- **Email** – The alarm notifies IAM Identity Center users with an email message.
- **Text** – The alarm notifies IAM Identity Center users with an SMS message.

b. For **Sender**, choose the sender.

> **Important**
>
> You must verify the sender email address in Amazon Simple Email Service (Amazon SES). For more information, see Verifying email addresses in Amazon SES, in the *Amazon Simple Email Service Developer Guide*.

c. Choose the attribute in **Recipient attribute - *optional***. The alarm uses the default value of the attribute that you choose.

d. Choose the attribute in **Custom message attribute - *optional***. The alarm uses the default value of the attribute that you choose.

9. In the **Instance** section, specify the **Default state** for this alarm. You can activate or deactivate this alarm for all assets that you create from this asset model in a later step.

10. In the **Advanced settings** settings, you can configure the permissions, the additional notification settings, the alarm state actions, the alarm model in SiteWise Monitor, and the acknowledge flow.

> **Note**
>
> AWS IoT Events alarms require the following service roles:
>
> - A role that AWS IoT Events assumes to send alarm state values to AWS IoT SiteWise.

- A role that AWS IoT Events assumes to send data to Lambda. You only need this role if your alarm sends notifications.

a. In the **Acknowledge flow** section, choose **Enabled** or **Disabled**. For more information about the acknowledge flow, see Alarm states.

b. In the **Permissions** section, do the following:

   i. For **AWS IoT Events role**, use an existing role or create a role with the required permissions. This role requires the `iotsitewise:BatchPutAssetPropertyValue` permission and a trust relationship that allows iotevents.amazonaws.com to assume the role.

   ii. For the **Lambda role**, use an existing role or create a role with the required permissions. This role requires the `lambda:InvokeFunction` and `sso-directory:DescribeUser` permissions and a trust relationship that allows `iotevents.amazonaws.com` to assume the role.

c. (Optional) In the **Additional notification settings** pane, do the following:

   - For **Manage your Lambda function**, do one of the following:
     - To have AWS IoT Events create a new Lambda function, choose **Create a new Lambda function**.
     - To use an existing Lambda function, choose **Use an existing Lambda function** and choose the name of the function.

     For more information, see Managing alarm notifications in the *AWS IoT Events Developer Guide*.

d. (Optional) In the **Set state action - *optional*** section, do the following:

   - Under **Alarm state actions**, add actions. and the choose **Save**.

     You can add up to 10 actions.

   AWS IoT Events can perform actions when the alarm is active. You can define built-in actions to use a timer or set a variable, or send data to other AWS resources. For more information, see Supported actions in the *AWS IoT Events Developer Guide*.

11. Choose **Create**.

> **Note**
>
> The AWS IoT Events console makes multiple API requests to add the alarm to the asset model. When you choose **Add alarm**, the console opens a dialog box that shows the progress of these API requests. Stay on this page until each API requests succeeds or until an API request fails. If a request fails, close the dialog box, fix the issue, and choose **Add alarm** to try again.

Defining an AWS IoT Events alarm (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to define an AWS IoT Events alarm that monitors an asset property. You can define the alarm on a new or existing asset model. After you define the alarm on the asset model, you create an alarm in AWS IoT Events and connect it to the asset model. In this process, you do the following:

Steps

- Step 1: Defining an alarm on an asset model
- Step 2: Defining an AWS IoT Events alarm model
- Step 3: Enabling data flow between AWS IoT SiteWise and AWS IoT Events

Step 1: Defining an alarm on an asset model

Add an alarm definition and associated properties to a new or existing asset model.

To define an alarm on an asset model (CLI)

1. Create a file called `asset-model-payload.json`. Follow the steps in these other sections to add your asset model's details to the file, but don't submit the request to create or update the asset model. In this section, you add an alarm definition to the asset model details in the `asset-model-payload.json` file.
   - For more information about how to create an asset model, see Creating an asset model (AWS CLI).
   - For more information about how to update an existing asset model, see Updating an asset or component model (AWS CLI).

> **Note**
>
> Your asset model must define at least one asset property, including the asset property to monitor with the alarm.

2. Add an alarm composite model (`assetModelCompositeModels`) to the asset model. An AWS IoT Events alarm composite model specifies the `IOT_EVENTS` type and specifies an alarm source property. You add the alarm source property after you create the alarm model in AWS IoT Events.

> **Important**
>
> The alarm composite model must have the same name as the AWS IoT Events alarm model you create later. Alarm model names can contain only alphanumeric characters. Specify a unique, alphanumeric name so that you can use the same name for the alarm model.

```
{
  ...
  "assetModelCompositeModels": [
    {
      "name": "BoilerTemperatureHighAlarm",
      "type": "AWS/ALARM",
      "properties": [
        {
          "name": "AWS/ALARM_TYPE",
          "dataType": "STRING",
          "type": {
            "attribute": {
              "defaultValue": "IOT_EVENTS"
            }
          }
        },
        {
          "name": "AWS/ALARM_STATE",
          "dataType": "STRUCT",
          "dataTypeSpec": "AWS/ALARM_STATE",
          "type": {
```

```
            "measurement": {}
          }
        }
      ]
    }
  ]
}
```

3. Add an alarm threshold attribute to the asset model. Specify the default value to use for this threshold. You can override this default value on each asset based on this model.

> **Note**
>
> The alarm threshold attribute must be an `INTEGER` or a `DOUBLE`.

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "Temperature Max Threshold",
      "dataType": "DOUBLE",
      "type": {
        "attribute": {
          "defaultValue": "105.0"
        }
      }
    }
  ]
}
```

4. (Optional) Add alarm notification attributes to the asset model. These attributes specify the IAM Identity Center recipient and other inputs that AWS IoT Events uses to send notifications when the alarm changes state. You can override these defaults on each asset based on this model.

> **Important**
>
> You can send alarm notifications to AWS IAM Identity Center users. To use this feature, you must enable IAM Identity Center. You can only enable IAM Identity Center in one

AWS Region at a time. This means that you can define alarm notifications only in the Region where you enable IAM Identity Center. For more information, see Getting started in the *AWS IAM Identity Center User Guide*.

Do the following:

a. Add an attribute that specifies the ID of your IAM Identity Center identity store. You can use the IAM Identity Center ListInstances API operation to list your identity stores. This operation works only in the Region where you enable IAM Identity Center.

```
aws sso-admin list-instances
```

Then, specify the identity store ID (for example, `d-123EXAMPLE`) as the default value for the attribute.

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "identityStoreId",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "d-123EXAMPLE"
        }
      }
    }
  ]
}
```

b. Add an attribute that specifies the ID of the IAM Identity Center user who receives notifications. To define a default notification recipient, add an IAM Identity Center user ID as the default value. Do one of the following to get an IAM Identity Center user ID:

   i. You can use the IAM Identity Center ListUsers API to get the ID of a user whose user name you know. Replace *d-123EXAMPLE* with the ID of your identity store, and replace *Name* with the user name of the user.

```
aws identitystore list-users \
  --identity-store-id d-123EXAMPLE \
  --filters AttributePath=UserName,AttributeValue=Name
```

ii. Use the IAM Identity Center console to browse your users and find a user ID.

Then, specify the user ID (for example, `123EXAMPLE-a1b2c3d4-5678-90ab-cdef-33333EXAMPLE`) as the default value for the attribute, or define the attribute without a default value.

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "userId",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "123EXAMPLE-a1b2c3d4-5678-90ab-cdef-33333EXAMPLE"
        }
      }
    }
  ]
}
```

c. (Optional) Add an attribute that specifies the default sender ID for SMS (text) message notifications. The sender ID displays as the message sender on messages that Amazon Simple Notification Service (Amazon SNS) sends. For more information, see Requesting sender IDs for SMS messaging with Amazon SNS in the *Amazon Simple Notification Service Developer Guide*.

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "senderId",
      "dataType": "STRING",
      "type": {
```

```
        "attribute": {
          "defaultValue": "MyFactory"
        }
      }
    }
  ]
}
```

d. (Optional) Add an attribute that specifies the default email address to use as the *from* address in email notifications.

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "fromAddress",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "my.factory@example.com"
        }
      }
    }
  ]
}
```

e. (Optional) Add an attribute that specifies the default subject to use in email notifications.

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "emailSubject",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "[ALERT] High boiler temperature"
        }
      }
    }
  ]
```

```
}
```

f. (Optional) Add an attribute that specifies an additional message to include in notifications. By default, notification messages include information about the alarm. You can also include an additional message that gives the user more information..

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "additionalMessage",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "Turn off the power before you check the alarm."
        }
      }
    }
  ]
}
```

5. Create the asset model or update the existing asset model. Do one of the following:

   - To create the asset model, run the following command.

     ```
     aws iotsitewise create-asset-model --cli-input-json file://asset-model-
     payload.json
     ```

   - To update the existing asset model, run the following command. Replace *asset-model-id* with the ID of the asset model.

     ```
     aws iotsitewise update-asset-model \
       --asset-model-id asset-model-id \
       --cli-input-json file://asset-model-payload.json
     ```

   After you run the command, note the `assetModelId` in the response.

Example: Boiler asset model

The following asset model represents a boiler that reports temperature data. This asset model defines an alarm that detects when the boiler overheats.

```
{
  "assetModelName": "Boiler Model",
  "assetModelDescription": "Represents a boiler.",
  "assetModelProperties": [
    {
      "name": "Temperature",
      "dataType": "DOUBLE",
      "unit": "C",
      "type": {
        "measurement": {}
      }
    },
    {
      "name": "Temperature Max Threshold",
      "dataType": "DOUBLE",
      "type": {
        "attribute": {
          "defaultValue": "105.0"
        }
      }
    },
    {
      "name": "identityStoreId",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "d-123EXAMPLE"
        }
      }
    },
    {
      "name": "userId",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "123EXAMPLE-a1b2c3d4-5678-90ab-cdef-33333EXAMPLE"
        }
      }
    },
    {
      "name": "senderId",
      "dataType": "STRING",
      "type": {
```

```
        "attribute": {
          "defaultValue": "MyFactory"
        }
      }
    },
    {
      "name": "fromAddress",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "my.factory@example.com"
        }
      }
    },
    {
      "name": "emailSubject",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "[ALERT] High boiler temperature"
        }
      }
    },
    {
      "name": "additionalMessage",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "Turn off the power before you check the alarm."
        }
      }
    }
  ],
  "assetModelHierarchies": [

  ],
  "assetModelCompositeModels": [
    {
      "name": "BoilerTemperatureHighAlarm",
      "type": "AWS/ALARM",
      "properties": [
        {
          "name": "AWS/ALARM_TYPE",
          "dataType": "STRING",
```

```
          "type": {
            "attribute": {
              "defaultValue": "IOT_EVENTS"
            }
          }
        },
        {
          "name": "AWS/ALARM_STATE",
          "dataType": "STRUCT",
          "dataTypeSpec": "AWS/ALARM_STATE",
          "type": {
            "measurement": {}
          }
        }
      ]
    }
  ]
}
```

Step 2: Defining an AWS IoT Events alarm model

Create the alarm model in AWS IoT Events. In AWS IoT Events, you use *expressions* to specify values in alarm models. You can use expressions to specify values from AWS IoT SiteWise to evaluate and use as inputs to the alarm. When AWS IoT SiteWise sends asset property values to the alarm model, AWS IoT Events evaluates the expression to get the value of the property or the ID of the asset. You can use the following expressions in the alarm model:

- **Asset property values**

  To get the value of an asset property, use the following expression. Replace *`assetModelId`* with the ID of the asset model and replace *`propertyId`* with the ID of the property.

  ```
  $sitewise.assetModel.`assetModelId`.`propertyId`.propertyValue.value
  ```

- **Asset IDs**

  To get the ID of the asset, use the following expression. Replace *`assetModelId`* with the ID of the asset model and replace *`propertyId`* with the ID of the property.

  ```
  $sitewise.assetModel.`assetModelId`.`propertyId`.assetId
  ```

> **Note**
>
> When you create the alarm model, you can define literals instead of expressions that evaluate to AWS IoT SiteWise values. This can reduce the number of attributes that you define on your asset model. However, if you define a value as a literal, you can't customize that value on assets based on the asset model. Your AWS IoT SiteWise Monitor users also can't customize the alarm, because they can configure alarm settings on assets only.

To create an AWS IoT Events alarm model (CLI)

1. When you create the alarm model in AWS IoT Events, you must specify the ID of each property that the alarm uses, which includes the following:

   - The alarm state property in the composite asset model
   - The property that the alarm monitors
   - The threshold attribute
   - (Optional) The IAM Identity Center identity store ID attribute
   - (Optional) The IAM Identity Center user ID attribute
   - (Optional) The SMS sender ID attribute
   - (Optional) The email *from* address attribute
   - (Optional) The email subject attribute
   - (Optional) The additional message attribute

   Run the following command to retrieve the IDs of these properties on the asset model. Replace *`asset-model-id`* with the ID of the asset model from the previous step.

   ```
   aws iotsitewise describe-asset-model --asset-model-id asset-model-id
   ```

   The operation returns a response that contains the asset model's details. Note the ID of each property that the alarm uses. You use these IDs when you create the AWS IoT Events alarm model in the next step.

2. Create the alarm model in AWS IoT Events. Do the following:

   a. Create a file called `alarm-model-payload.json`.

b. Copy the following JSON object into the file.

c. Enter a name (`alarmModelName`), description (`alarmModelDescription`), and severity (`severity`) for your alarm. For severity, specify an integer that reflects your company's severity levels.

> **Important**
>
> The alarm model must have the same name as the alarm composite model that you defined on your asset model earlier.
> Alarm model names can contain only alphanumeric characters.

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3
}
```

d. Add the comparison rule (`alarmRule`) to the alarm. This rule defines the property to monitor (`inputProperty`), the threshold value to compare (`threshold`), and the comparison operator to use (`comparisonOperator`).

- Replace *`assetModelId`* with the ID of the asset model.
- Replace *`alarmPropertyId`* with the ID of the property that the alarm monitors.
- Replace *`thresholdAttributeId`* with the ID of the threshold attribute property.
- Replace *`GREATER`* with the operator to use to compare the property values with the threshold. Choose from the following options:
  - `LESS`
  - `LESS_OR_EQUAL`
  - `EQUAL`
  - `NOT_EQUAL`
  - `GREATER_OR_EQUAL`
  - `GREATER`

```
{
```

```
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
 "$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  }
}
```

e. Add an action (`alarmEventActions`) to send alarm state to the AWS IoT SiteWise when the alarm changes state.

> **Note**
>
> For advanced configuration, you can define additional actions to perform when the alarm changes state. For example, you might call an AWS Lambda function or publish to an MQTT topic. For more information, see Working with other AWS services in the *AWS IoT Events Developer Guide*.

- Replace *assetModelId* with the ID of the asset model.
- Replace *alarmPropertyId* with the ID of the property that the alarm monitors.
- Replace *alarmStatePropertyId* with the ID of the alarm state property in the alarm composite model.

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
```

```
      "threshold":
 "$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
          "propertyId": "'alarmStatePropertyId'"
        }
      }
    ]
  }
}
```

f. (Optional) Configure alarm notification settings. The alarm notification action uses a Lambda function in your account to send alarm notifications. For more information, see Requirements for alarm notifications. In the alarm notification settings, you can configure SMS and email notifications to send to IAM Identity Center users. Do the following:

i. Add the alarm notification configuration (`alarmNotification`) to the payload in `alarm-model-payload.json`.

- Replace *alarmNotificationFunctionArn* with the ARN of the Lambda function that handles alarm notifications.

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
 "$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
```

```
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
          "propertyId": "'alarmStatePropertyId'"
        }
      }
    ]
  },
  "alarmNotification": {
    "notificationActions": [
      {
        "action": {
          "lambdaAction": {
            "functionArn": "alarmNotificationFunctionArn"
          }
        }
      }
    ]
  }
}
```

ii. (Optional) Configure the SMS notifications (`smsConfigurations`) to send to an IAM Identity Center user when the alarm changes state.

- Replace *`identityStoreIdAttributeId`* with the ID of the attribute that contains the ID of the IAM Identity Center identity store.
- Replace *`userIdAttributeId`* with the ID of the attribute that contains the ID of the IAM Identity Center user.
- Replace *`senderIdAttributeId`* with the ID of the attribute that contains the Amazon SNS sender ID, or remove `senderId` from the payload.
- Replace *`additionalMessageAttributeId`* with the ID of the attribute that contains the additional message, or remove `additionalMessage` from the payload.

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
```

```
 "alarmRule": {
   "simpleRule": {
     "inputProperty":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
     "comparisonOperator": "GREATER",
     "threshold":
"$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
   }
 },
 "alarmEventActions": {
   "alarmActions": [
     {
       "iotSiteWise": {
         "assetId":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
         "propertyId": "'alarmStatePropertyId'"
       }
     }
   ]
 },
 "alarmNotification": {
   "notificationActions": [
     {
       "action": {
         "lambdaAction": {
           "functionArn": "alarmNotificationFunctionArn"
         }
       },
       "smsConfigurations": [
         {
           "recipients": [
             {
               "ssoIdentity": {
                 "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.va
                 "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
               }
             }
           ],
           "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId`.propertyValue.value",
           "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.
```

```
          }
        ]
      }
    ]
  }
}
```

iii. (Optional) Configure the email notifications (`emailConfigurations`) to send to an IAM Identity Center user when the alarm changes state.

- Replace *`identityStoreIdAttributeId`* with the ID of the IAM Identity Center identity store ID attribute property.
- Replace *`userIdAttributeId`* with the ID of the IAM Identity Center user ID attribute property.
- Replace *`fromAddressAttributeId`* with the ID of the "from" address attribute property, or remove `from` from the payload.
- Replace *`emailSubjectAttributeId`* with the ID of the email subject attribute property, or remove `subject` from the payload.
- Replace *`additionalMessageAttributeId`* with the ID of the additional message attribute property, or remove `additionalMessage` from the payload.

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
 "$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
```

```
          "propertyId": "'alarmStatePropertyId'"
        }
      }
    ]
  },
  "alarmNotification": {
    "notificationActions": [
      {
        "action": {
          "lambdaAction": {
            "functionArn": "alarmNotificationFunctionArn"
          }
        },
        "smsConfigurations": [
          {
            "recipients": [
              {
                "ssoIdentity": {
                  "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.va
                  "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                }
              }
            ],
            "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId`.propertyValue.value",
            "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.
          }
        ],
        "emailConfigurations": [
          {
            "from":
"$sitewise.assetModel.`assetModelId`.`fromAddressAttributeId`.propertyValue.value"
            "recipients": {
              "to": [
                {
                  "ssoIdentity": {
                    "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.va
                    "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                  }
```

```
                }
              ]
            },
            "content": {
              "subject":
 "$sitewise.assetModel.`assetModelId`.`emailSubjectAttributeId`.propertyValue.value
              "additionalMessage":
 "$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.
            }
          }
        ]
      }
    ]
  }
}
```

g. (Optional) Add the alarm capabilities (`alarmCapabilities`) to the payload in `alarm-model-payload.json`. In this object, you can specify if the acknowledge flow is enabled and the default enable state for assets based on the asset model. For more information about the acknowledge flow, see Alarm states.

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
 "$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
          "propertyId": "'alarmStatePropertyId'"
        }
      }
```

```
    ]
  },
  "alarmNotification": {
    "notificationActions": [
      {
        "action": {
          "lambdaAction": {
            "functionArn": "alarmNotificationFunctionArn"
          }
        },
        "smsConfigurations": [
          {
            "recipients": [
              {
                "ssoIdentity": {
                  "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value"
                  "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                }
              }
            ],
            "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId`.propertyValue.value",
            "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.valu
          }
        ],
        "emailConfigurations": [
          {
            "from":
"$sitewise.assetModel.`assetModelId`.`fromAddressAttributeId`.propertyValue.value",
            "recipients": {
              "to": [
                {
                  "ssoIdentity": {
                    "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value"
                    "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                  }
                }
              ]
            },
```

```
            "content": {
              "subject":
 "$sitewise.assetModel.`assetModelId`.`emailSubjectAttributeId`.propertyValue.value",
              "additionalMessage":
 "$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.valu
            }
          }
        ]
      }
    ]
  },
  "alarmCapabilities": {
    "initializationConfiguration": {
      "disabledOnInitialization": false
    },
    "acknowledgeFlow": {
      "enabled": true
    }
  }
}
```

h. Add the IAM service role (`roleArn`) that AWS IoT Events can assume to send data to AWS IoT SiteWise. This role requires the `iotsitewise:BatchPutAssetPropertyValue` permission and a trust relationship that allows `iotevents.amazonaws.com` to assume the role. To send notifications, this role also requires the `lambda:InvokeFunction` and `sso-directory:DescribeUser` permissions. For more information, see Alarm service roles in the *AWS IoT Events Developer Guide*.

- Replace the `roleArn` with the ARN of the role that AWS IoT Events can assume to perform these actions.

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
```

```
      "threshold":
"$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
          "propertyId": "'alarmStatePropertyId'"
        }
      }
    ]
  },
  "alarmNotification": {
    "notificationActions": [
      {
        "action": {
          "lambdaAction": {
            "functionArn": "alarmNotificationFunctionArn"
          }
        },
        "smsConfigurations": [
          {
            "recipients": [
              {
                "ssoIdentity": {
                  "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value"
                  "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                }
              }
            ],
            "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId`.propertyValue.value",
            "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.valu
          }
        ],
        "emailConfigurations": [
          {
```

```
            "from":
 "$sitewise.assetModel.`assetModelId`.`fromAddressAttributeId`.propertyValue.value",
            "recipients": {
              "to": [
                {
                  "ssoIdentity": {
                    "identityStoreId":
 "$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value"
                    "userId":
 "$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                  }
                }
              ]
            },
            "content": {
              "subject":
 "$sitewise.assetModel.`assetModelId`.`emailSubjectAttributeId`.propertyValue.value",
              "additionalMessage":
 "$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.valu
            }
          }
        ]
      }
    ]
  },
  "alarmCapabilities": {
    "initializationConfiguration": {
      "disabledOnInitialization": false
    },
    "acknowledgeFlow": {
      "enabled": false
    }
  },
  "roleArn": "arn:aws:iam::123456789012:role/MyIoTEventsAlarmRole"
}
```

i. Run the following command to create the AWS IoT Events alarm model from the payload in `alarm-model-payload.json`.

```
aws iotevents create-alarm-model --cli-input-json file://alarm-model-
payload.json
```

j. The operation returns a response that includes the ARN of the alarm model, `alarmModelArn`. Copy this ARN to set in the alarm definition on your asset model in the next step.

Step 3: Enabling data flow between AWS IoT SiteWise and AWS IoT Events

After you create the required resources in AWS IoT SiteWise and AWS IoT Events, you can enable data flow between the resources to enable your alarm. In this section, you update the alarm definition in the asset model to use the alarm model that you created in the previous step.

To enable data flow between AWS IoT SiteWise and AWS IoT Events (CLI)

- Set the alarm model as the alarm's source in the asset model. Do the following:

  a. Run the following command to retrieve the existing asset model definition. Replace *`asset-model-id`* with the ID of the asset model.

  ```
  aws iotsitewise describe-asset-model --asset-model-id asset-model-id
  ```

  The operation returns a response that contains the asset model's details.

  b. Create a file called `update-asset-model-payload.json` and copy the previous command's response into the file.

  c. Remove the following key-value pairs from the `update-asset-model-payload.json` file:

  - `assetModelId`
  - `assetModelArn`
  - `assetModelCreationDate`
  - `assetModelLastUpdateDate`
  - `assetModelStatus`

  d. Add the alarm source property (`AWS/ALARM_SOURCE`) to the alarm composite model that you defined earlier. Replace *`alarmModelArn`* with the ARN of the alarm model, which sets the value of the alarm source property.

  ```
  {
    ...
    "assetModelCompositeModels": [
  ```

```
    ...
    {
      "name": "BoilerTemperatureHighAlarm",
      "type": "AWS/ALARM",
      "properties": [
        {
          "id": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
          "name": "AWS/ALARM_TYPE",
          "dataType": "STRING",
          "type": {
            "attribute": {
              "defaultValue": "IOT_EVENTS"
            }
          }
        },
        {
          "id": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
          "name": "AWS/ALARM_STATE",
          "dataType": "STRUCT",
          "dataTypeSpec": "AWS/ALARM_STATE",
          "type": {
            "measurement": {}
          }
        },
        {
          "name": "AWS/ALARM_SOURCE",
          "dataType": "STRING",
          "type": {
            "attribute": {
              "defaultValue": "alarmModelArn"
            }
          }
        }
      ]
    }
  ]
}
```

e. Run the following command to update the asset model with the definition stored in the `update-asset-model-payload.json` file. Replace *`asset-model-id`* with the ID of the asset model.

```
aws iotsitewise update-asset-model \
```

```
    --asset-model-id asset-model-id \
    --cli-input-json file://update-asset-model-payload.json
```

Your asset model now defines an alarm that detects in AWS IoT Events. The alarm monitors the target property in all assets based on this asset model. You can configure the alarm on each asset to customize properties such as the threshold or IAM Identity Center recipient for each asset. For more information, see Configuring alarms on assets.

Defining external alarms

External alarms contain the state of an alarm that you detect outside of AWS IoT SiteWise.

Defining an external alarm (console)

You can use the AWS IoT SiteWise console to define an external alarm on an existing asset model. To define an external alarm on a new asset model, create the asset model, and then complete these steps. For more information, see Creating asset models.

To define an alarm on an asset model

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Models**.
3. Choose the asset model for which to define an alarm.
4. Choose the **Alarm definitions** tab.
5. Choose **Add alarm**.
6. In **Alarm type options**, choose **External alarm**.
7. Enter a name for your alarm.
8. (Optional) Enter a description for your alarm.
9. Choose **Add alarm**.

Defining an external alarm (CLI)

You can use the AWS CLI to define an external alarm on a new or existing asset model.

To add an external alarm to an asset model, you add an alarm composite model to the asset model. An external alarm composite model specifies the `EXTERNAL` type and doesn't specify an alarm source property. The following example composite alarm defines an external temperature alarm.

```
{
  ...
  "assetModelCompositeModels": [
    {
      "name": "BoilerTemperatureHighAlarm",
      "type": "AWS/ALARM",
      "properties": [
        {
          "name": "AWS/ALARM_TYPE",
          "dataType": "STRING",
          "type": {
            "attribute": {
              "defaultValue": "EXTERNAL"
            }
          }
        },
        {
          "name": "AWS/ALARM_STATE",
          "dataType": "STRUCT",
          "dataTypeSpec": "AWS/ALARM_STATE",
          "type": {
            "measurement": {}
          }
        }
      ]
    }
  ]
}
```

For more information about how to add a composite model to a new or existing asset model, see the following:

- Creating an asset model (AWS CLI)
- Updating an asset or component model (AWS CLI)

After you define the external alarm, you can ingest alarm state to assets based on the asset model. For more information, see Ingesting external alarm state.

Configuring alarms on assets

After you define an AWS IoT Events alarm on an asset model, you can configure the alarm on each asset based on the asset model. You can edit the threshold value and the notification settings for the alarm. Each of these values is an attribute on the asset, so you can update the default value of the attribute to configure these values.

> **Note**
>
> You can configure these values for AWS IoT Events alarms, but not on external alarms.

Topics

Configuring a threshold value (console)

You can use the AWS IoT SiteWise console to update the value of the attribute that specifies the threshold value of an alarm.

To update an alarm's threshold value (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the asset for which you want to update an alarm threshold value.

   > **Tip**
   >
   > You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. Choose **Edit**.
5. Find the attribute that the alarm uses for its threshold value, and then enter its new value.
6. Choose **Save**.

Configuring a threshold value (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to update the value of the attribute that specifies the threshold value of an alarm.

You must know your asset's `assetId` and property's `propertyId` to complete this procedure. You can also use the external ID. If you created an asset and don't know its `assetId`, use the ListAssets API to list all the assets for a specific model. Use the DescribeAsset operation to view your asset's properties including property IDs.

Use the BatchPutAssetPropertyValue operation to assign attribute values to your asset. You can use this operation to set multiple attributes at once. This operation's payload contains a list of entries, and each entry contains the asset ID, property ID, and attribute value.

To update an attribute's value (AWS CLI)

1. Create a file called `batch-put-payload.json` and copy the following JSON object into the file. This example payload demonstrates how to set a wind turbine's latitude and longitude. Update the IDs, values, and timestamps to modify the payload for your use case.

```
{
  "entries": [
    {
      "entryId": "windfarm3-turbine7-latitude",
      "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "propertyId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
      "propertyValues": [
        {
          "value": {
            "doubleValue": 47.6204
          },
          "timestamp": {
            "timeInSeconds": 1575691200
          }
        }
      ]
    },
    {
      "entryId": "windfarm3-turbine7-longitude",
      "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "propertyId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE",
      "propertyValues": [
```

```
        {
          "value": {
            "doubleValue": 122.3491
          },
          "timestamp": {
            "timeInSeconds": 1575691200
          }
        }
      ]
    }
  ]
}
```

- Each entry in the payload contains an `entryId` that you can define as any unique string. If any request entries fail, each error will contain the `entryId` of the corresponding request so that you know which requests to retry.
- To set an attribute value, you can include one timestamp-quality-value (TQV) structure in the list of `propertyValues` for each attribute property. This structure must contain the new `value` and the current `timestamp`.
  - `value` – A structure that contains one of the following fields, depending on the type of the property being set:
    - `booleanValue`
    - `doubleValue`
    - `integerValue`
    - `stringValue`
  - `timestamp` – A structure that contains the current Unix epoch time in seconds, `timeInSeconds`. AWS IoT SiteWise rejects any data points with timestamps that existed longer than 7 days in the past or newer than 5 minutes in the future.

For more information about how to prepare a payload for BatchPutAssetPropertyValue, see Ingesting data using the AWS IoT SiteWise API.

2. Run the following command to send the attribute values to AWS IoT SiteWise:

```
aws iotsitewise batch-put-asset-property-value -\-cli-input-json file://batch-put-payload.json
```

Configuring notification settings (console)

You can use the AWS IoT SiteWise console to update the value of the attributes that specify the notification settings for an alarm.

To update an alarm's notification settings (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the asset for which you want to update the alarm settings.
4. Choose **Edit**.
5. Find the attribute that the alarm uses for the notification setting that you want to change, and then enter its new value.
6. Choose **Save**.

Configuring notification settings (CLI)

You can use the AWS Command Line Interface (AWS CLI) to update the value of the attribute that specifies the notification settings for an alarm.

You must know your asset's `assetId` and property's `propertyId` to complete this procedure. You can also use the external ID. If you created an asset and don't know its `assetId`, use the ListAssets API to list all the assets for a specific model. Use the DescribeAsset operation to view your asset's properties including property IDs.

Use the BatchPutAssetPropertyValue operation to assign attribute values to your asset. You can use this operation to set multiple attributes at once. This operation's payload contains a list of entries, and each entry contains the asset ID, property ID, and attribute value.

To update an attribute's value (AWS CLI)

1. Create a file called `batch-put-payload.json` and copy the following JSON object into the file. This example payload demonstrates how to set a wind turbine's latitude and longitude. Update the IDs, values, and timestamps to modify the payload for your use case.

```
{
  "entries": [
    {
```

```
      "entryId": "windfarm3-turbine7-latitude",
      "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "propertyId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
      "propertyValues": [
        {
          "value": {
            "doubleValue": 47.6204
          },
          "timestamp": {
            "timeInSeconds": 1575691200
          }
        }
      ]
    },
    {
      "entryId": "windfarm3-turbine7-longitude",
      "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "propertyId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE",
      "propertyValues": [
        {
          "value": {
            "doubleValue": 122.3491
          },
          "timestamp": {
            "timeInSeconds": 1575691200
          }
        }
      ]
    }
  ]
}
```

- Each entry in the payload contains an `entryId` that you can define as any unique string. If any request entries fail, each error will contain the `entryId` of the corresponding request so that you know which requests to retry.
- To set an attribute value, you can include one timestamp-quality-value (TQV) structure in the list of `propertyValues` for each attribute property. This structure must contain the new `value` and the current `timestamp`.
  - `value` – A structure that contains one of the following fields, depending on the type of the property being set:
    - `booleanValue`

    - `doubleValue`
    - `integerValue`
    - `stringValue`
  - `timestamp` – A structure that contains the current Unix epoch time in seconds, `timeInSeconds`. AWS IoT SiteWise rejects any data points with timestamps that existed longer than 7 days in the past or newer than 5 minutes in the future.

  For more information about how to prepare a payload for BatchPutAssetPropertyValue, see Ingesting data using the AWS IoT SiteWise API.
2. Run the following command to send the attribute values to AWS IoT SiteWise:

```
aws iotsitewise batch-put-asset-property-value -\-cli-input-json file://batch-put-payload.json
```

Responding to alarms

When an AWS IoT Events alarm changes state, you can do the following to respond to the alarm:

- Acknowledge an alarm to indicate that you are handling the issue.
- Snooze an alarm to disable it temporarily.
- Disable an alarm to disable it permanently until you enable it again.
- Enable a disabled alarm to detect alarm state.
- Reset an alarm to clear its state and latest value.

You can use the AWS IoT SiteWise console or the AWS IoT Events API to respond to an alarm.

> **Note**
>
> You can respond to AWS IoT Events alarms, but not external alarms.

Topics

- Responding to an alarm (console)
- Responding to an alarm (API)

Responding to an alarm (console)

You can use the AWS IoT SiteWise console to acknowledge, snooze, disable, or enable an alarm.

Topics

Acknowledge an alarm (console)

You can acknowledge an alarm to indicate that you're handling the issue.

> **Note**
>
> You must enable the acknowledge flow on the alarm so that you can acknowledge the alarm. This option is enabled by default if you define the alarm from the AWS IoT SiteWise console.

To acknowledge an alarm (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the asset to for which you want to acknowledge an alarm.

   > **Tip**
   >
   > You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. Choose the **Alarms** tab.
5. Select the alarm to acknowledge, and then choose **Actions** to open the response action menu.
6. Choose **Acknowledge**. The alarm's state changes to **Acknowledged**.

Snooze an alarm (console)

You can snooze an alarm to disable it temporarily. Specify the duration for which to snooze the alarm.

To snooze an alarm (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the asset to for which you want to snooze an alarm.

> **Tip**
>
> You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. Choose the **Alarms** tab.
5. Select the alarm to snooze, and then choose **Actions** to open the response action menu.
6. Choose **Snooze**. A model opens where you specify the duration to snooze.
7. Choose the **Snooze length** or enter a **Custom snooze length**.
8. Choose **Save**. The alarm's state changes to **Snoozed**.

Disable an alarm (console)

You can disable an alarm so that it doesn't detect anymore. After you disable the alarm, you must enable it again if you want the alarm to detect.

To disable an alarm (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the asset to for which you want to disable an alarm.

> **Tip**
>
> You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. Choose the **Alarms** tab.

5. Select the alarm to disable, and then choose **Actions** to open the response action menu.
6. Choose **Disable**. The alarm's state changes to **Disabled**.

Enable an alarm (console)

You can enable an alarm to detect again after you disable or snooze it.

To enable an alarm (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the asset to for which you want to enable an alarm.

> **Tip**
>
> You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. Choose the **Alarms** tab.
5. Select the alarm to enable, and then choose **Actions** to open the response action menu.
6. Choose **Enable**. The alarm's state changes to **Normal**.

Reset an alarm (console)

You can reset an alarm to clear its state and latest value.

To reset an alarm (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the asset to for which you want to reset an alarm.

> **Tip**
>
> You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. Choose the **Alarms** tab.
5. Select the alarm to enable, and then choose **Actions** to open the response action menu.

6. Choose **Reset**. The alarm's state changes to **Normal**.

Responding to an alarm (API)

You can use the AWS IoT Events API to acknowledge, snooze, disable, enable, or reset an alarm. For more information, see the following operations in the *AWS IoT Events API Reference*:

- BatchAcknowledgeAlarm
- BatchSnoozeAlarm
- BatchDisableAlarm
- BatchEnableAlarm
- BatchResetAlarm

For more information, see Responding to alarms in the *AWS IoT Events Developer Guide*.

Ingesting external alarm state

External alarms are alarms that you evaluate outside of AWS IoT SiteWise. You can use external alarms when you have a data source that reports alarm state that you want to ingest to AWS IoT SiteWise.

Alarm state properties require a specific format for alarm state data values. Each data value must be a JSON object serialized to a string. Then, you ingest the serialized string as a string value. For more information, see Alarm state properties.

Example Example alarm state data value (not serialized)

```
{
  "stateName": "Active"
}
```

Example Example alarm state data value (serialized)

```
{\"stateName\":\"Active\"}
```

> **Note**
>
> If your data source can't report data in this format, or you can't convert your data to this format before you ingest it, you might choose not to use an alarm property. Instead, you can ingest the data as a measurement property with the string data type, for example. For more information, see Defining data streams from equipment (measurements) and Ingesting data to AWS IoT SiteWise.

Mapping external alarm state streams

You can define property aliases to map your data streams to your alarm state properties. This helps you easily identify an alarm state property when you ingest or retrieve data. For more information about property aliases, see Mapping industrial data streams to asset properties.

Topics

- Mapping external alarm state streams (console)
- Mapping external alarm state streams (AWS CLI)

Mapping external alarm state streams (console)

You can define property aliases to map your data streams to your alarm state properties. This helps you easily identify an alarm state property when you ingest or retrieve data. For more information about property aliases, see Mapping industrial data streams to asset properties.

You can use the AWS IoT SiteWise console to set an alias for an alarm state property.

To set a property alias for an alarm state property (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the asset for which you want to set a property alias.

   > **Tip**
   >
   > You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. Choose **Alarms**.

5. Select the external alarm for which you want to set a property alias.
6. Choose **View**.
7. In the **Alarm state details** pane, choose **Edit**.
8. Enter the property alias.
9. Choose **Update**.

Mapping external alarm state streams (AWS CLI)

You can define property aliases to map your data streams to your alarm state properties. This helps you easily identify an alarm state property when you ingest or retrieve data. For more information about property aliases, see Mapping industrial data streams to asset properties.

You can use the AWS Command Line Interface (AWS CLI) to set an alias for an alarm state property.

You must know your asset's `assetId` and property's `propertyId` to complete this procedure. You can also use the external ID. If you created an asset and don't know its `assetId`, use the ListAssets API to list all the assets for a specific model. Use the DescribeAsset operation to view your asset's properties including property IDs.

> **Note**
>
> The DescribeAsset response includes the list of composite asset models for the asset. Each alarm is a composite model. To find the `propertyId`, find the composite model for the alarm, and then find the `AWS/ALARM_STATE` property in that composite model.

For more information about how to set the property alias, see Setting a property alias (AWS CLI).

Ingesting alarm state data

Alarm state properties expect alarm state as a serialized JSON string. To ingest alarm state to an external alarm in AWS IoT SiteWise, you ingest this serialized string as a timestamped string value. The following example demonstrates a state data value for an active alarm.

```
{\"stateName\":\"Active\"}
```

To identify an alarm state property, you can specify one of the following:

- The `assetId` and `propertyId` of the alarm property that you're sending data to.
- The `propertyAlias`, which is a data stream alias (for example, `/company/windfarm/3/turbine/7/temperature/high`). To use this option, you must first set your alarm property's alias. To learn how to set property aliases for alarm state properties, see Mapping external alarm state streams.

The following example BatchPutAssetPropertyValue API payload demonstrates how to format the state of an external alarm. This external alarm reports when a wind turbine's rotations per minute (RPM) reading is too high.

Example Example BatchPutAssetPropertyValue payload for alarm state data

```
{
    "entries": [
      {
        "entryId": "unique entry ID",
        "propertyAlias": "/company/windfarm/3/turbine/7/temperature/high",
        "propertyValues": [
          {
            "value": {
              "stringValue": "{\"stateName\":\"Active\"}"
            },
            "timestamp": {
              "timeInSeconds": 1607550262
            }
          }
        ]
      }
    ]
  }
```

For more information about how to use the `BatchPutAssetPropertyValue` API to ingest data, see Ingesting data using the AWS IoT SiteWise API.

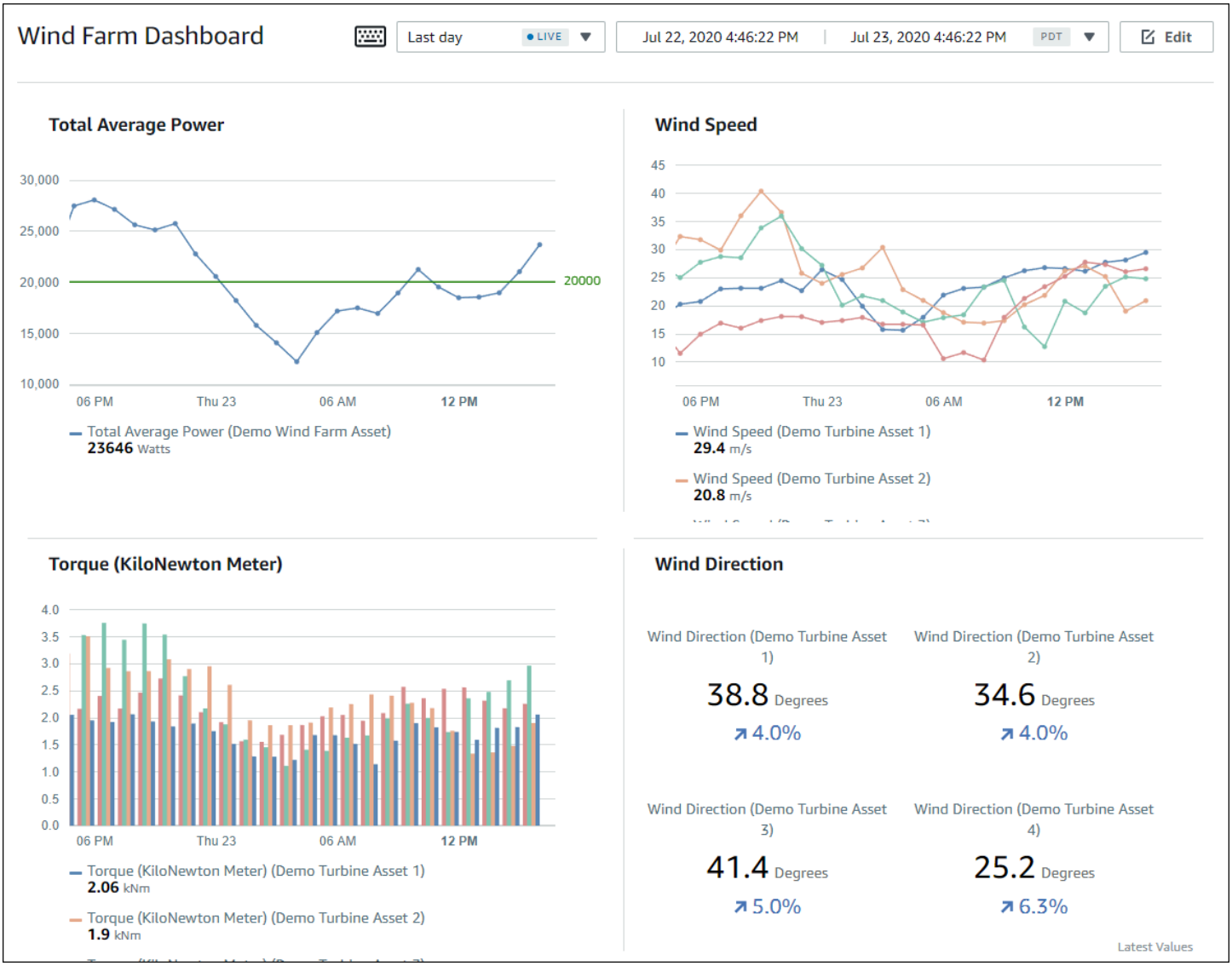
For more information about other ways to ingest data, see Ingesting data to AWS IoT SiteWise.

Monitoring data with AWS IoT SiteWise Monitor

You can use AWS IoT SiteWise to monitor the data from your processes, devices, and equipment by creating SiteWise Monitor web portals. SiteWise Monitor is a feature of AWS IoT SiteWise that you can use to create portals in the form of a managed web application. You can then use these portals to view and share your operational data. You can create projects with dashboards to visualize data from your processes, devices, and equipment that are connected to AWS IoT.

Domain experts, such as process engineers, can use these portals to quickly get insights into their operational data to understand device and equipment behavior.

The following is an example dashboard that displays data for a wind farm.



Because AWS IoT SiteWise captures data over time, you can use SiteWise Monitor to view operational data over time, or the last reported values at specific points in time. This lets you uncover insights that might otherwise be difficult to find.

SiteWise Monitor roles

Four roles interact with SiteWise Monitor:

AWS administrator

The AWS administrator uses the AWS IoT SiteWise console to create portals. The AWS administrator can also assign portal administrators and add portal users. Portal administrators later assign portal users to projects as owners or viewers. The AWS administrator works exclusively in the AWS console.

Portal administrator

Each SiteWise Monitor portal has one or more portal administrators. Portal administrators use the portal to create projects that contain collections of assets and dashboards. The portal administrator then assigns assets and owners to each project. By controlling access to the project, portal administrators specify which assets that project owners and viewers can see.

Project owner

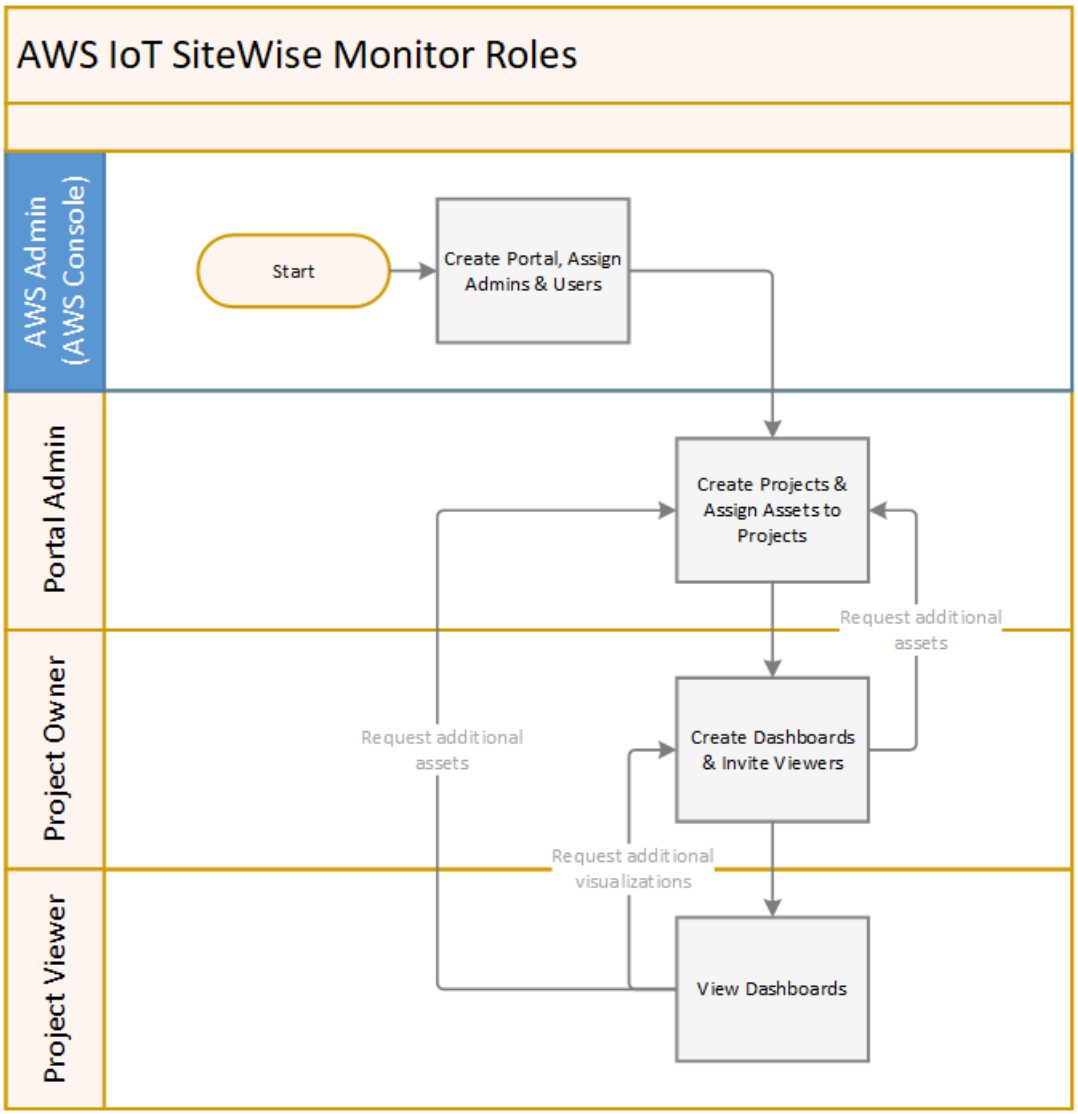
Each SiteWise Monitor project has owners. Project owners create visualizations in the form of dashboards to represent operational data in a consistent manner. When dashboards are ready to share, the project owner can invite viewers to the project. Project owners can also assign other owners to the project. Project owners can configure thresholds and notification settings for alarms.

Project viewer

Each SiteWise Monitor project has viewers. Project viewers can connect to the portal to view the dashboards that project owners created. In each dashboard, project viewers can adjust the time range to better understand operational data. Project viewers can only view dashboards in the projects to which they have access. Project viewers can acknowledge and snooze alarms.

Depending on your organization, the same person might perform multiple roles.

The following image illustrates how these four roles interact in the SiteWise Monitor portal.



You can manage who has access to your data by using AWS IAM Identity Center or IAM. Your data users can sign in to SiteWise Monitor from a desktop or mobile browser using their IAM Identity Center or IAM credentials.

SAML federation

IAM Identity Center and IAM support identity federation with SAML (Security Assertion Markup Language) 2.0. SAML 2.0 is an open standard that many external identity providers (IdPs) use to authenticate users and pass their identity and security information to service providers (SPs). SPs are typically applications or services. SAML federation enables your SiteWise Monitor portal administrators and users to sign in to their assigned portals with external credentials, such as their corporate usernames and passwords.

You can configure IAM Identity Center and IAM to use SAML-based federation for access to your SiteWise Monitor portals.

IAM Identity Center

Your portal administrators and users can sign in to the AWS access portal with their corporate usernames and passwords. They can then navigate to their assigned SiteWise Monitor portals. IAM Identity Center uses certificates to set up a SAML trust relationship between your identity provider and AWS. For more information, SCIM profile and SAML 2.0 implementation in the *AWS IAM Identity Center User Guide*.

IAM

Your portal administrators and users can request temporary security credentials to access their assigned SiteWise Monitor portals. You create a SAML identity provider identity in IAM to set up a trust relationship between your identity provider and AWS. For more information, see Using SAML-based federation for API access to AWS, in the *IAM User Guide*.

Your portal administrators and users can sign in to your company's portal and select the option to go to the AWS Management console. They can then navigate to their assigned SiteWise Monitor portals. Your company's portal handles the exchange of trust between your identity provider and AWS. For more information, see Enabling SAML 2.0 federated users to access the AWS Management Console in the *IAM User Guide*.

> **Note**
>
> When adding users or administrators to the portal, avoid creating IAM policies that restrict user permissions, such as limited IP. Any attached policies with restricted permissions will not be able to connect to the AWS IoT SiteWise portal.

SiteWise Monitor concepts

To use SiteWise Monitor, you should be familiar with the following concepts:

Portal

An AWS IoT SiteWise Monitor portal is a web application that you can use to visualize and share your AWS IoT SiteWise data. A portal has one or more administrators and contains zero or more projects.

Project

Each SiteWise Monitor portal contains a set of projects. Each project has a subset of your AWS IoT SiteWise assets associated with it. Project owners create one or more dashboards to provide a consistent way to view the data associated with those assets. Project owners can invite viewers to the project to allow them to view the assets and dashboards in the project. The project is the basic unit of sharing within SiteWise Monitor. Project owners can invite users who were given access to the portal by the AWS administrator. A user must have access to a portal before a project in that portal can be shared with that user.

Asset

When data is ingested into AWS IoT SiteWise from your industrial equipment, your devices, equipment, and processes are each represented as assets. Each asset has properties and alarms associated with it. The portal administrator assigns sets of assets to each project.

Property

Properties are time series data associated with assets. For example, a piece of equipment might have a serial number, a location, a make and model, and an install date. It might also have time series values for availability, performance, quality, temperature, pressure, and so on.

Alarm

Alarms monitor properties to identify when equipment is outside of its operating range. Each alarm defines a threshold and a property to monitor. When the property exceeds the threshold, the alarm becomes active and indicates that you or someone on your team should address the issue. Project owners can customize the thresholds and notification settings for alarms. Project viewers can acknowledge and snooze alarms, and they can leave a message with details about the alarm or the action that they took to address it.

Dashboard

Each project contains a set of dashboards. Dashboards provide a set of visualizations for the values of a set of assets. Project owners create the dashboards and the visualizations that it contains. When a project owner is ready to share the set of dashboards, the owner can invite viewers to the project, which gives them access to all dashboards in the project. If you want a different set of viewers for different dashboards, you must divide the dashboards between projects. When viewers look at dashboards, they can customize time range to look at specific data.

Visualization

In each dashboard, project owners decide how to display the properties and alarms of the assets associated with the project. Availability might be represented as a line chart, while other values might be displayed as bar charts or key performance indicators (KPIs). Alarms are best displayed as status grids and status timelines. Project owners customize each visualization to provide the best understanding of the data for that asset.

Getting started with AWS IoT SiteWise Monitor

If you're the AWS administrator for your organization, you create portals from the AWS IoT SiteWise console. Complete the following steps to create a portal so that members of your organization can view your AWS IoT SiteWise data:

1. Configure and create a portal
2. Add portal administrators and send invitation emails
3. Add portal users

After you create a portal, the portal administrator can view your AWS IoT SiteWise assets and assign them to projects in the portal. Project owners can then create dashboards to visualize the properties of the assets that help project viewers understand how your devices, processes, and equipment are performing.

> **Note**
>
> When adding users or administrators to the portal, avoid creating AWS Identity and Access Management (IAM) policies that restrict user permissions, such as limited IP. Any attached policies with restricted permissions will not be able to connect to the AWS IoT SiteWise portal.

You can follow a tutorial that walks through the steps required to set up a portal with a project, dashboards, and multiple users for a specific scenario using wind farm data. For more information, see Visualizing and sharing wind farm data in SiteWise Monitor.

Topics

- Creating a portal

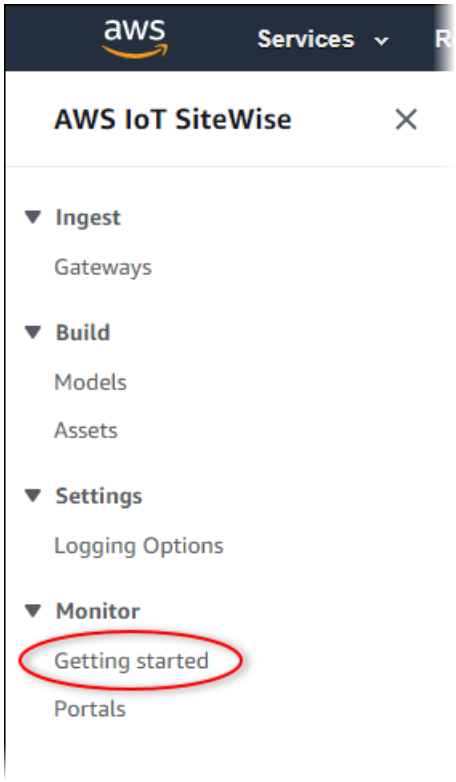
- Configuring your portal
- Inviting administrators
- Adding portal users

Creating a portal

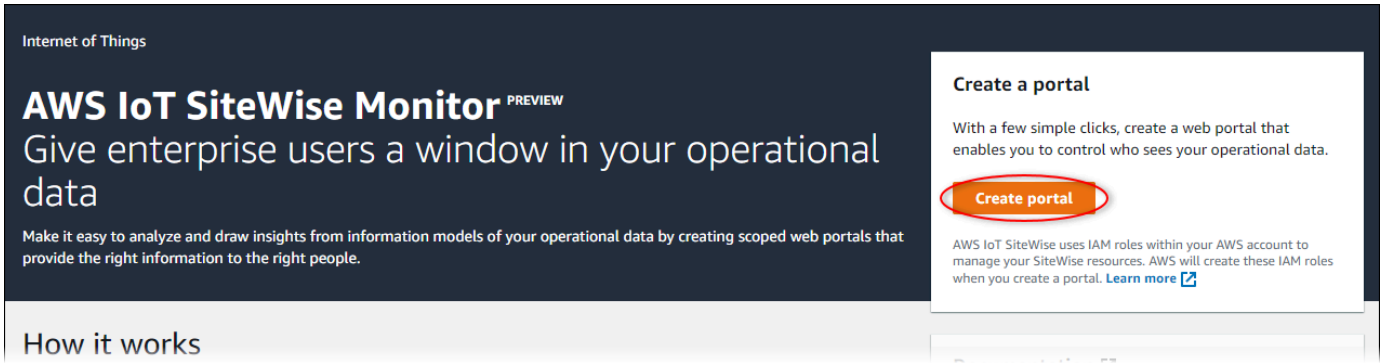
You create a SiteWise Monitor portal in the AWS IoT SiteWise console.

To create a portal

1. Sign in to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Monitor**, **Getting started**.



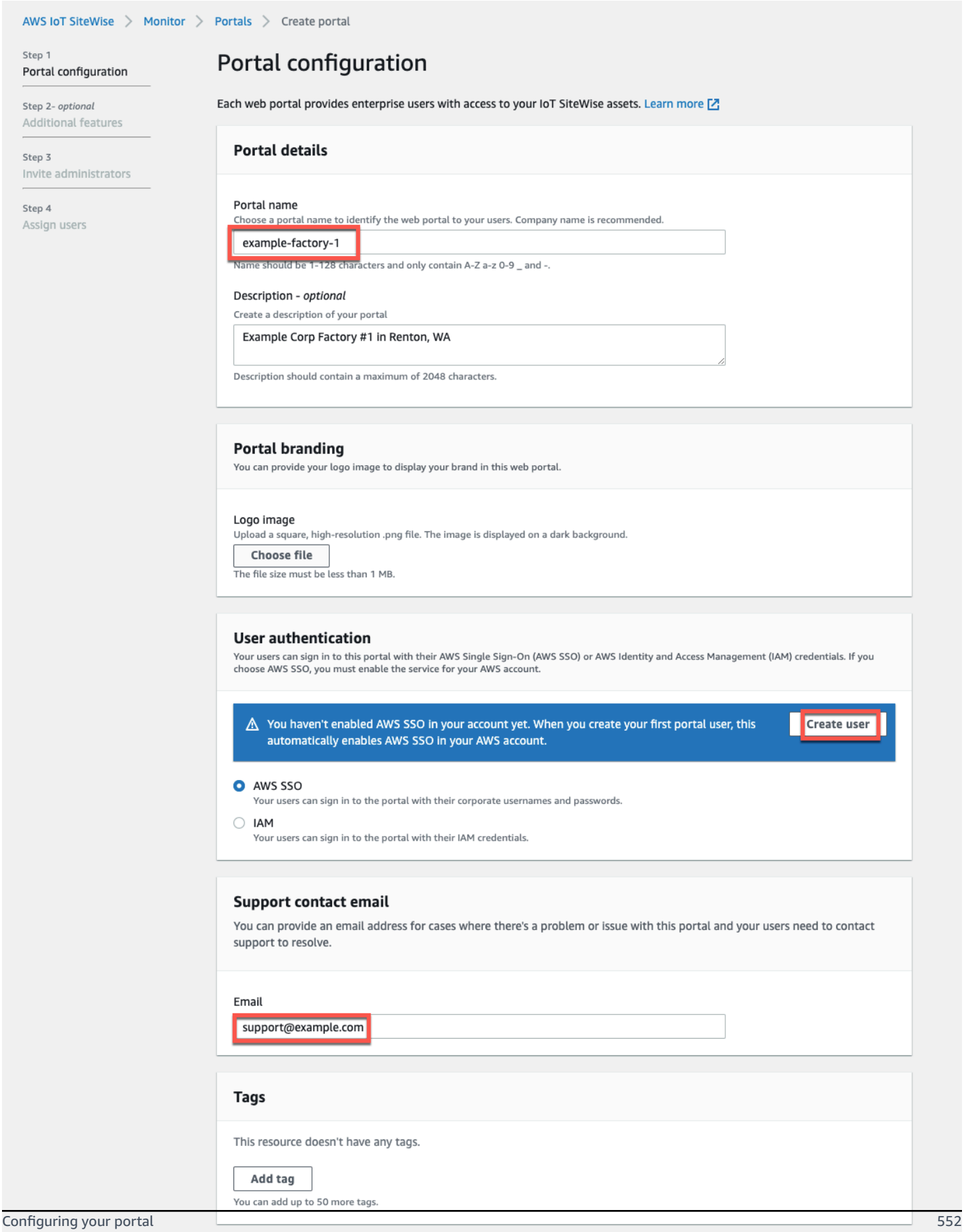
3. Choose **Create Portal**.



Next, you must provide some basic information to configure your portal.

Configuring your portal

Your users use portals to view your data. You can customize a portal's name, description, branding, user authentication, support contact email, and permissions.


AWS IoT SiteWise > Monitor > Portals > Create portal
Step 1
Portal configuration
Step 2- optional
Additional features
Step 3
Invite administrators
Step 4
Assign users
Portal configuration
Each web portal provides enterprise users with access to your IoT SiteWise assets. Learn more
Portal details
Portal name
Choose a portal name to identify the web portal to your users. Company name is recommended.
example-factory-1
Name should be 1-128 characters and only contain A-Z a-z 0-9 _ and -.
Description - optional
Create a description of your portal
Example Corp Factory #1 in Renton, WA
Description should contain a maximum of 2048 characters.
Portal branding
You can provide your logo image to display your brand in this web portal.
Logo image
Upload a square, high-resolution .png file. The image is displayed on a dark background.
Choose file
The file size must be less than 1 MB.
User authentication
Your users can sign in to this portal with their AWS Single Sign-On (AWS SSO) or AWS Identity and Access Management (IAM) credentials. If you choose AWS SSO, you must enable the service for your AWS account.
You haven't enabled AWS SSO in your account yet. When you create your first portal user, this automatically enables AWS SSO in your AWS account.
Create user
AWS SSO
Your users can sign in to the portal with their corporate usernames and passwords.
IAM
Your users can sign in to the portal with their IAM credentials.
Support contact email
You can provide an email address for cases where there's a problem or issue with this portal and your users need to contact support to resolve.
Email
support@example.com
Tags
This resource doesn't have any tags.
Add tag
You can add up to 50 more tags.
Permissions
SiteWise Monitor assumes this role to give permissions to your federated users to access AWS IoT SiteWise resources. Learn

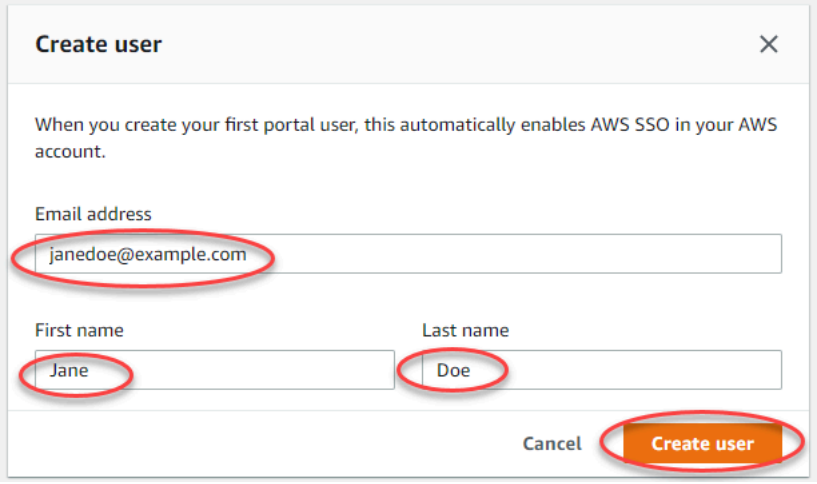
To configure a portal

1. Enter a name for your portal.
2. (Optional) Enter a description for your portal. If you have multiple portals, use meaningful descriptions to help you keep track of what each portal contains.
3. (Optional) Upload an image to display your brand in the portal. Choose a square, PNG image. If you upload a non-square image, the portal scales the image down to a square.
4. Choose one of the following options:
   - Choose **IAM Identity Center** if your portal users sign in to this portal with their corporate user names and passwords.

     If you haven't enabled IAM Identity Center in your account, do the following:

     a. Choose **Create user**.

     b. On the **Create user** page, to create the first portal, enter the user's email address, first name, and last name, and then choose **Create user**.



> **Note**
>
> - AWS automatically enables IAM Identity Center in your account when you create the first portal user.
> - You can configure IAM Identity Center in only one Region at a time. SiteWise Monitor connects to the Region that you configured for IAM Identity Center. This means that you use one Region for IAM Identity Center access, but you can create portals in any Region.

- Choose **IAM** if your portal users sign in to this portal with their IAM credentials.

> **Important**
>
> Users or roles must have the `iotsitewise:DescribePortal` permission to sign in to the portal.

5. Enter an email address that portal users can contact when they have an issue with the portal and need help to resolve it.
6. (Optional) Add tags for your portal. For more information, see Tagging your AWS IoT SiteWise resources.
7. Choose one of the following options:
   - Choose **Create and use a new service role**. By default, SiteWise Monitor automatically creates a service role for each portal. This role allows your portal users to access your AWS IoT SiteWise resources. For more information, see Using service roles for AWS IoT SiteWise Monitor.
   - Choose **Use an existing service role**, and then choose the target role.
8. Choose **Next**
9. (Optional) Enable alarms for your portal. For more information, see Enabling alarms for your portals.
10. Choose **Create**. AWS IoT SiteWise will create your portal.

> **Note**
>
> If you close the console, you can finish the setup process by adding administrators and users. For more information, see Adding or removing portal administrators. If you don't want to keep this portal, delete it so it doesn't use resources. For more information, see Deleting a portal.

The **Status** column can be one of the following values.

- **CREATING** - AWS IoT SiteWise is processing your request to create the portal. This process can take several minutes to complete.

- **UPDATING** - AWS IoT SiteWise is processing your request to update the portal. This process can take several minutes to complete.
- **PENDING** - AWS IoT SiteWise is waiting for the DNS record propagation to finish. This process can take several minutes to complete. You can delete the portal while the status is **PENDING**.
- **DELETING** - AWS IoT SiteWise is processing your request to delete the portal. This process can take several minutes to complete.
- **ACTIVE** - When the portal becomes active, your portal users can access it.
- **FAILED** - AWS IoT SiteWise couldn't process your request to create, update, or delete the portal. If you enabled AWS IoT SiteWise to send logs to Amazon CloudWatch Logs, you can use these logs to troubleshoot issues. For more information, see Monitoring AWS IoT SiteWise with CloudWatch Logs.

A message appears when your portal is created.

Next, you must invite one or more portal administrators to the portal. So far, you created a portal but no one can access it.

Inviting administrators

To get started in your new portal, you must assign a portal administrator. The portal administrator creates projects, chooses project owners, and assigns assets to projects. Portal administrators can see all of your AWS IoT SiteWise assets.

Based on the user authentication service, choose one of the following options:

IAM Identity Center

If you're using SiteWise Monitor for the first time, you can choose the user that you created earlier to be the portal administrator. If you want to add another user as a portal administrator, you can create an IAM Identity Center user from this page. Alternatively, you can connect an external identity provider to IAM Identity Center. For more information, see the AWS IAM Identity Center User Guide.

To invite administrators

1. Select the check boxes for the users that you want as your portal administrators. This adds the users to the **Portal administrators** list.

> **Note**
>
> If you use IAM Identity Center as your identity store, and you're signed in to your AWS Organizations management account, you can choose **Create user** to create an IAM Identity Center user. IAM Identity Center sends the new user an email for them to set their password. You can then assign the user to the portal as an administrator. For more information, see Manage identities in IAM Identity Center.

2. (Optional) Choose **Send invite to selected users**. Your email client opens, and an invitation is populated in the message body.

   You can customize the email before you send it to your portal administrators. You can also send the email to your portal administrators later. If you're trying SiteWise Monitor for the first time and adding your new IAM Identity Center or IAM user or role as the portal administrator, you don't need to email yourself.
3. If you add a user that you don't want as an administrator, clear the check box for that user.
4. When you're finished inviting portal administrators, choose **Next**.

IAM

You can choose a user or role to be the portal administrator. If you want to add another user or role as a portal administrator, you can create a user or role in the IAM console. For more information, see Creating an IAM user in your AWS account and Creating IAM roles in the *IAM User Guide*.

To invite administrators

1. Do the following:
   - Choose **IAM users** to add an IAM user as your portal administrator.
   - Choose **IAM roles** to add an IAM role as your portal administrator.
2. Select the check boxes for the users or roles that you want as your portal administrators. This adds the users or roles to the **Portal administrators** list.
3. If you add a user or role that you don't want as an administrator, clear the check box for that user or role.
4. When you're finished inviting portal administrators, choose **Next**.

> **Important**
>
> Users or roles must have the `iotsitewise:DescribePortal` permission to sign in to the portal.

> **Note**
>
> If you use IAM Identity Center as your identity store, and you're signed in to your AWS Organizations management account, you can choose **Create user** to create an IAM Identity Center user. IAM Identity Center sends the new user an email for them to set their password. You can then assign the user to the portal as an administrator. For more information, see Manage identities in IAM Identity Center.

You can change the list of portal administrators later. For more information, see Adding or removing portal administrators.

> **Note**
>
> Because only a portal administrator can create projects and assign assets to them, you should specify at least one portal administrator.

As the last step, you add users who can access your new portal.

Adding portal users

You control which users have access to your portals. In each portal, the portal administrators create one or more projects and assign portal users as owners or viewers for each project. Each project owner can invite additional portal users to own or view the project.

Based on the user authentication service, choose one of the following options:

IAM Identity Center

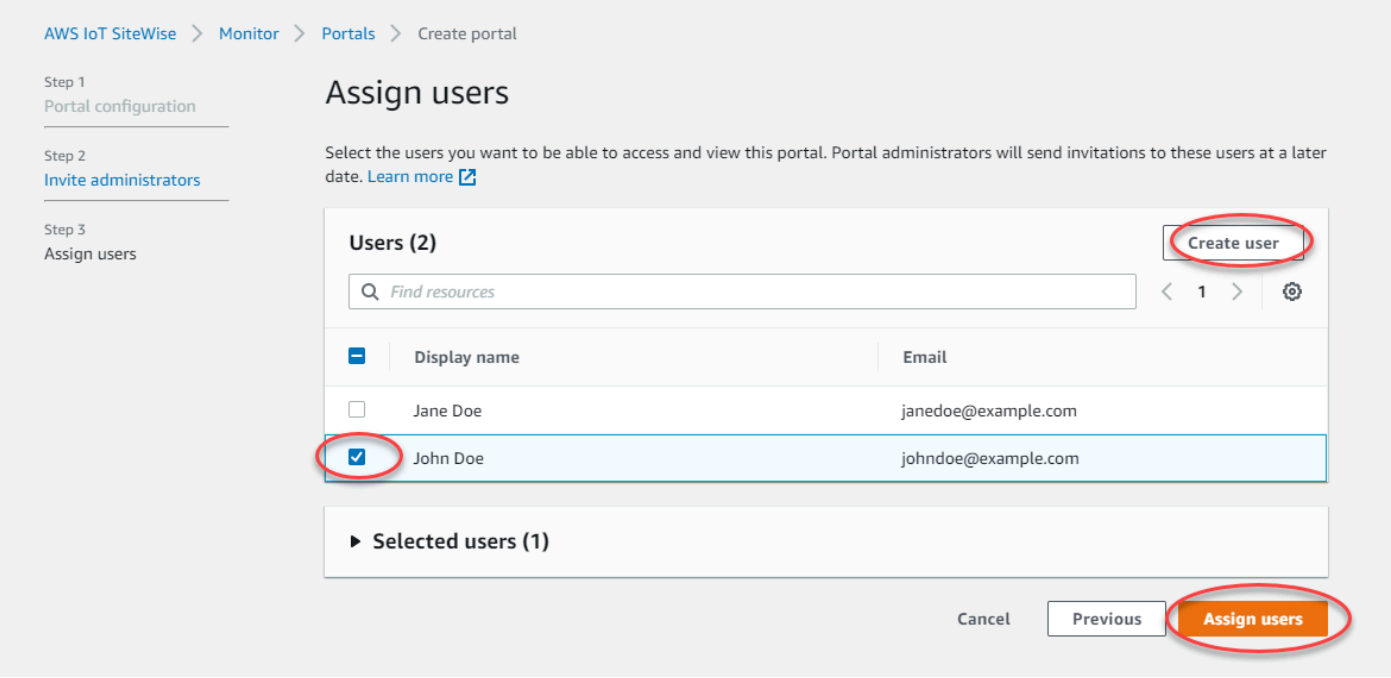
If you want to add a user to the **Users** list, complete the following steps.

To add portal users

1. Choose users from the **Users** list to add to the portal. This adds the users to the **Portal users** list. If you're using SiteWise Monitor for the first time, you don't need to add your portal administrator as a portal user.

   > **Note**
   >
   > If you use IAM Identity Center as your identity store, and you're signed in to your AWS Organizations management account, you can choose **Create user** to create an IAM Identity Center user. IAM Identity Center sends the new user an email for them to set their password. You can then assign the user to the portal as a user. For more information, see Manage identities in IAM Identity Center.

2. If you add a user that you don't want to have access to the portal, clear the check box for that user.
3. When you're finished selecting users, choose **Assign users**.



IAM

If you see the user or role that you want to add in the **IAM users** or **IAM roles** list, complete the following steps.

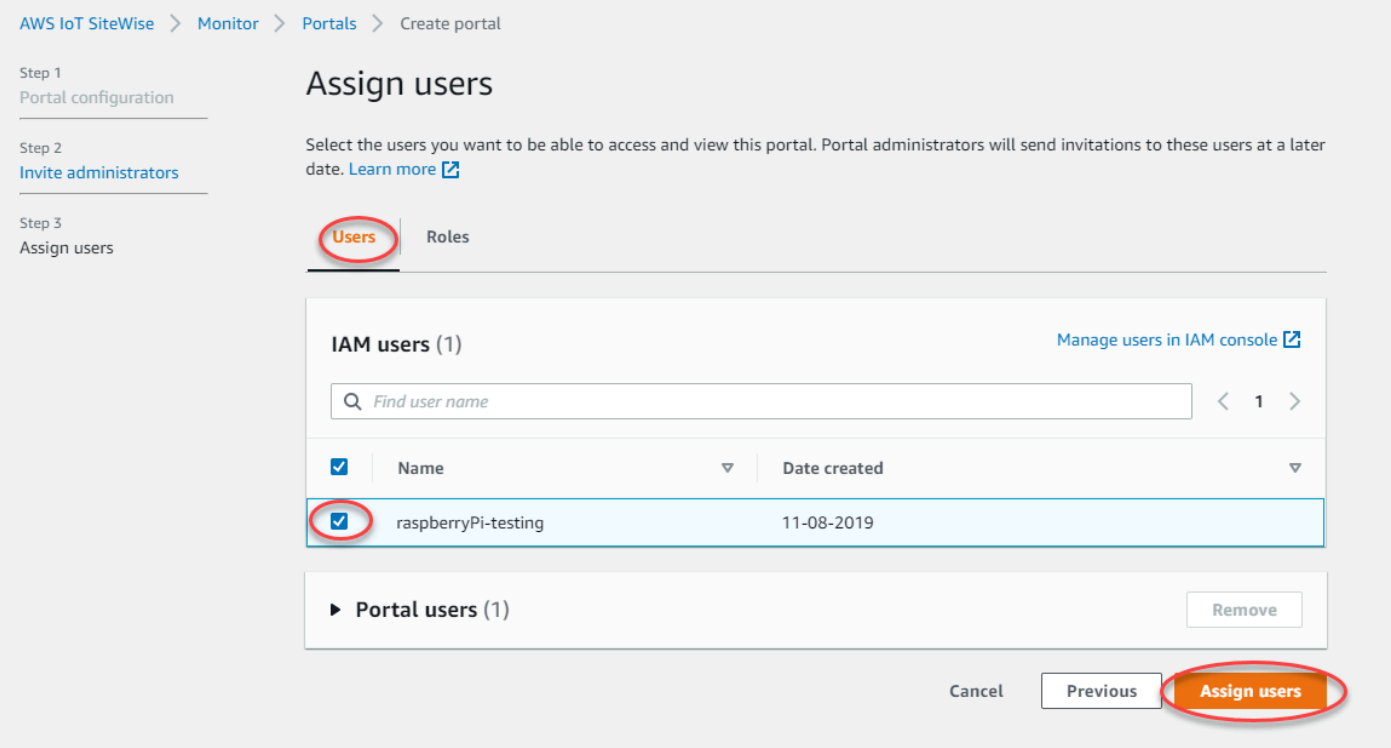
To add portal users

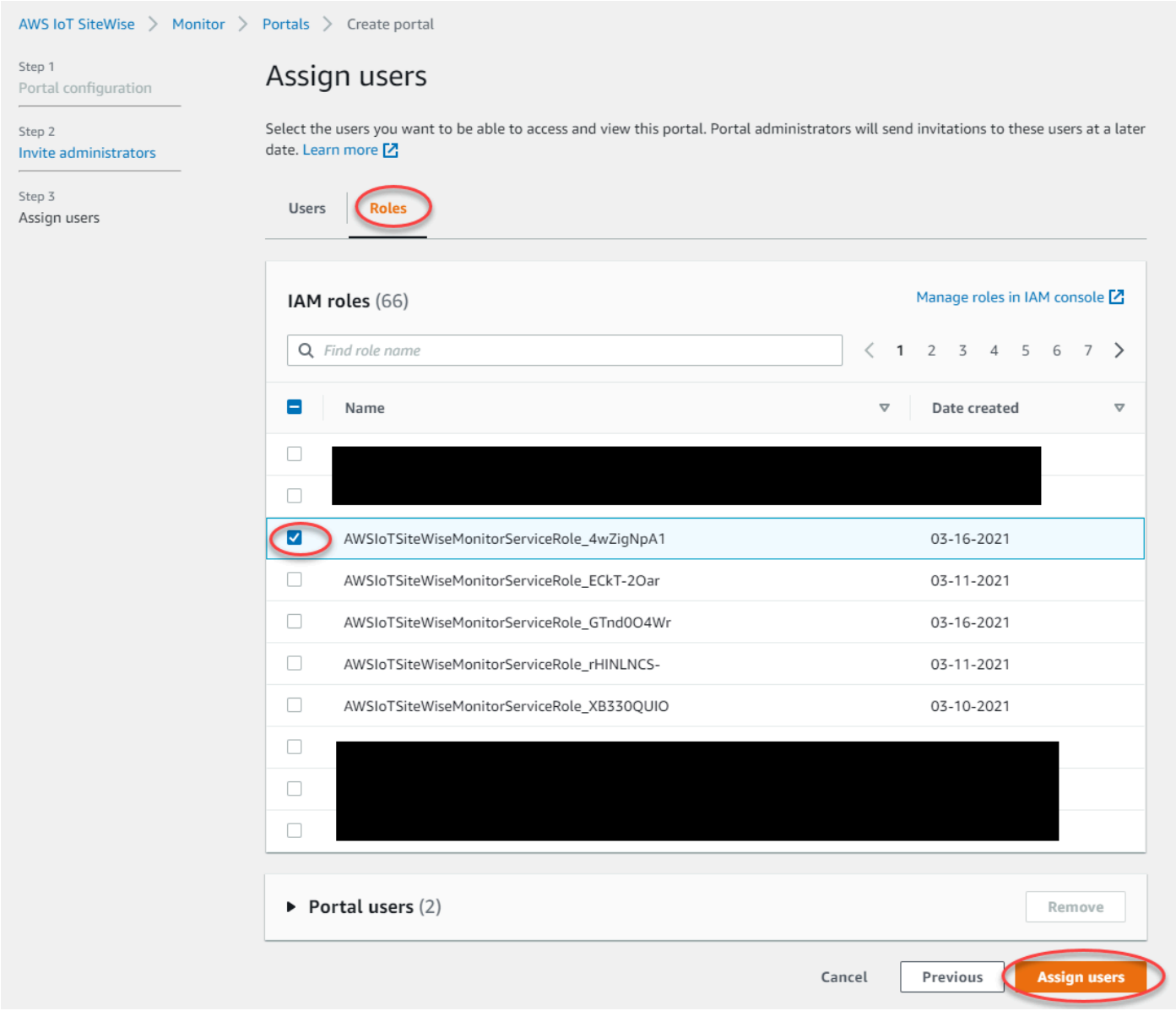
1. Do the following options:

   - Choose **IAM users** to add an IAM user as a portal user.
   - Choose **IAM roles** to add an IAM role as a portal user.

   If you're using SiteWise Monitor for the first time, you don't need to add your portal administrator as a portal user.
2. Select the check boxes for the users or roles that you want as portal users. This adds the users or roles to the **Portal users** list.
3. If you add a user that you don't want to have access to the portal, clear the check box for that user.
4. When you're finished selecting users, choose **Assign users**.

> **Important**
>
> Users or roles must have the `iotsitewise:DescribePortal` permission to sign in to the portal.


AWS IoT SiteWise > Monitor > Portals > Create portal
Step 1
Portal configuration
Step 2
Invite administrators
Step 3
Assign users
Assign users
Select the users you want to be able to access and view this portal. Portal administrators will send invitations to these users at a later date. Learn more
Users
Roles
IAM users (1)
Manage users in IAM console
Find user name
1
Name
Date created
raspberryPi-testing
11-08-2019
Portal users (1)
Remove
Cancel
Previous
Assign users



Congratulations! You successfully created a portal, assigned portal administrators, and assigned users who can use that portal when invited to do so. Your portal administrators can now create projects and add assets to those projects. Then, your project owners can create dashboards to visualize the data for each project's assets.

You can change the list of portal users later. For more information, see Adding or removing portal users.

If you need to make changes to the portal, see Administering your SiteWise Monitor portals.

To get started in the portal, see Getting started in the *SiteWise Monitor Application Guide*.

Creating dashboards (AWS Command Line Interface)

When you define visualizations (or widgets) in dashboards using the AWS CLI, you must specify the following information in the `dashboardDefinition` JSON document. This definition is a parameter of the CreateDashboard and UpdateDashboard operations.

`widgets`

A list of widget definition structures that each contain the following information:

`type`

The type of widget. AWS IoT SiteWise provides the following widget types:

- `sc-line-chart` – A line chart. For more information, see Line charts in the *AWS IoT SiteWise Monitor Application Guide*.
- `sc-scatter-chart` – A scatter chart. For more information, see Scatter charts in the *AWS IoT SiteWise Monitor Application Guide*.
- `sc-bar-chart` – A bar chart. For more information, see Bar charts in the *AWS IoT SiteWise Monitor Application Guide*.
- `sc-status-grid` – A status widget that shows the latest value of asset properties as a grid. For more information, see Status widgets in the *AWS IoT SiteWise Monitor Application Guide*.
- `sc-status-timeline` – A status widget that shows the historical values of asset properties as a timeline. For more information, see Status widgets in the *AWS IoT SiteWise Monitor Application Guide*.
- `sc-kpi` – A key performance indicator (KPI) visualization. For more information, see KPI widgets in the *AWS IoT SiteWise Monitor Application Guide*.
- `sc-table` – A table widget. For more information, see Table widgets in the *AWS IoT SiteWise Monitor Application Guide*.

`title`

The title of the widget.

`x`

The horizontal position of the widget, starting from the left of the grid. This value refers to the widget's position in the dashboard's grid.

`y`

The vertical position of the widget, starting from the top of the grid. This value refers to the widget's position in the dashboard's grid.

`width`

The width of the widget, expressed in number of spaces on the dashboard's grid.

`height`

The height of the widget, expressed in number of spaces on the dashboard's grid.

`metrics`

A list of metric structures that each define a data stream for this widget. Each structure in the list must contain the following information:

`label`

A label to display for this metric.

`type`

The type of data source for this metric. AWS IoT SiteWise provides the following metric types:

- `iotsitewise` – The dashboard fetches data for an asset property in AWS IoT SiteWise. If you choose this option, you must define `assetId` and `propertyId` for this metric.

`assetId`

(Optional) The ID of an asset in AWS IoT SiteWise.

This field is required if you choose `iotsitewise` for `type` in this metric.

`propertyId`

(Optional) The ID of an asset property in AWS IoT SiteWise.

This field is required if you choose `iotsitewise` for `type` in this metric.

`analysis`

(Optional) A structure that defines the analysis, such as trend lines, to display for the widget. For more information, see Configuring trend lines in the *AWS IoT SiteWise Monitor Application Guide*. You can add one of each type of trend line per property in the widget. The analysis structure contains the following information:

`trends`

(Optional) A list of trend structures that each define a trend analysis for this widget. Each structure in the list contains the following information:

`type`

The type of trend line. Choose the following option:

- `linear-regression` – Display a linear regression line. SiteWise Monitor uses the least squares method to calculate the linear regression.

`annotations`

(Optional) An annotations structure that defines thresholds for the widget. For more information, see Configuring thresholds in the *AWS IoT SiteWise Monitor Application Guide*. You can add up to six annotations per widget. The annotations structure contains the following information:

`y`

(Optional) A list of annotation structures that each define a horizontal threshold for this widget. Each structure in the list contains the following information:

`comparisonOperator`

The comparison operator for the threshold. Choose one of the following:

- `LT` – Highlight properties that have at least one data point less than the `value`.
- `GT` – Highlight properties that have at least one data point greater than the `value`.
- `LTE` – Highlight properties that have at least one data point less than or equal to the `value`.
- `GTE` – Highlight properties that have at least one data point greater than or equal to the `value`.
- `EQ` – Highlight properties that have at least one data point equal to the `value`.

`value`

The threshold value to compare data points with the `comparisonOperator`.

`color`

(Optional) The 6-digit hexadecimal code of the threshold color. The visualization displays property legends in this color for properties with at least one data point that meets the threshold rule. Defaults to black (`#000000`).

`showValue`

(Optional) Whether or not to show the value of the threshold in the margins of the widget. Defaults to `true`.

`properties`

(Optional) A flat dictionary of properties for the widget. The members of this structure are context-dependent. AWS IoT SiteWise provides the following widgets that use `properties`:

- Line charts, scatter charts, and bar charts have the following property:

  `colorDataAcrossThresholds`

  (Optional) Whether or not to change the color of the data that crosses the thresholds in this widget. When you enable this option, the data that crosses a threshold appears in the color that you choose. Defaults to `true`.

- Status grids have the following property:

  `labels`

  (Optional) A structure that defines the labels to display on the status grid. The labels structure contains the following information:

  `showValue`

  (Optional) Whether or not to display the unit and value for each asset property in this widget. Defaults to `true`.

Example Example dashboard definition

The following example defines a dashboard from a payload stored in a JSON file.

```
aws iotsitewise create-dashboard \
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeeEXAMPLE \
```

```
  --dashboard-name "Wind Farm Dashboard" \
  --dashboard-definition file://dashboard-definition.json
```

The following JSON example for `dashboard-definition.json` defines dashboard with the following visualization widgets:

- A line chart that visualizes total wind farm power in the upper left of the dashboard. This line chart includes a threshold that indicates when the wind farm outputs less power than its minimum expected output. This line chart also includes a linear regression trend line.
- A bar chart that visualizes wind speed for four turbines in the upper right of the dashboard.

> **Note**
>
> This example represents line and bar chart visualizations on a dashboard. This dashboard is similar to the example wind farm dashboard.

```
{
  "widgets": [
    {
      "type": "sc-line-chart",
      "title": "Total Average Power",
      "x": 0,
      "y": 0,
      "height": 3,
      "width": 3,
      "metrics": [
        {
          "label": "Power",
          "type": "iotsitewise",
          "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
          "propertyId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
          "analysis": {
            "trends": [
              {
                "type": "linear-regression"
              }
            ]
          }
        }
```

```
      ],
      "annotations": {
        "y": [
          {
            "comparisonOperator": "LT",
            "value": 20000,
            "color": "#D13212",
            "showValue": true
          }
        ]
      }
    },
    {
      "type": "sc-bar-chart",
      "title": "Wind Speed",
      "x": 3,
      "y": 3,
      "height": 3,
      "width": 3,
      "metrics": [
        {
          "label": "Turbine 1",
          "type": "iotsitewise",
          "assetId": "a1b2c3d4-5678-90ab-cdef-2a2a2EXAMPLE",
          "propertyId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE"
        },
        {
          "label": "Turbine 2",
          "type": "iotsitewise",
          "assetId": "a1b2c3d4-5678-90ab-cdef-2b2b2EXAMPLE",
          "propertyId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE"
        },
        {
          "label": "Turbine 3",
          "type": "iotsitewise",
          "assetId": "a1b2c3d4-5678-90ab-cdef-2c2c2EXAMPLE",
          "propertyId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE"
        },
        {
          "label": "Turbine 4",
          "type": "iotsitewise",
          "assetId": "a1b2c3d4-5678-90ab-cdef-2d2d2EXAMPLE",
          "propertyId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE"
        }
```

```
      ]
    }
  ]
}
```

Enabling alarms for your portals

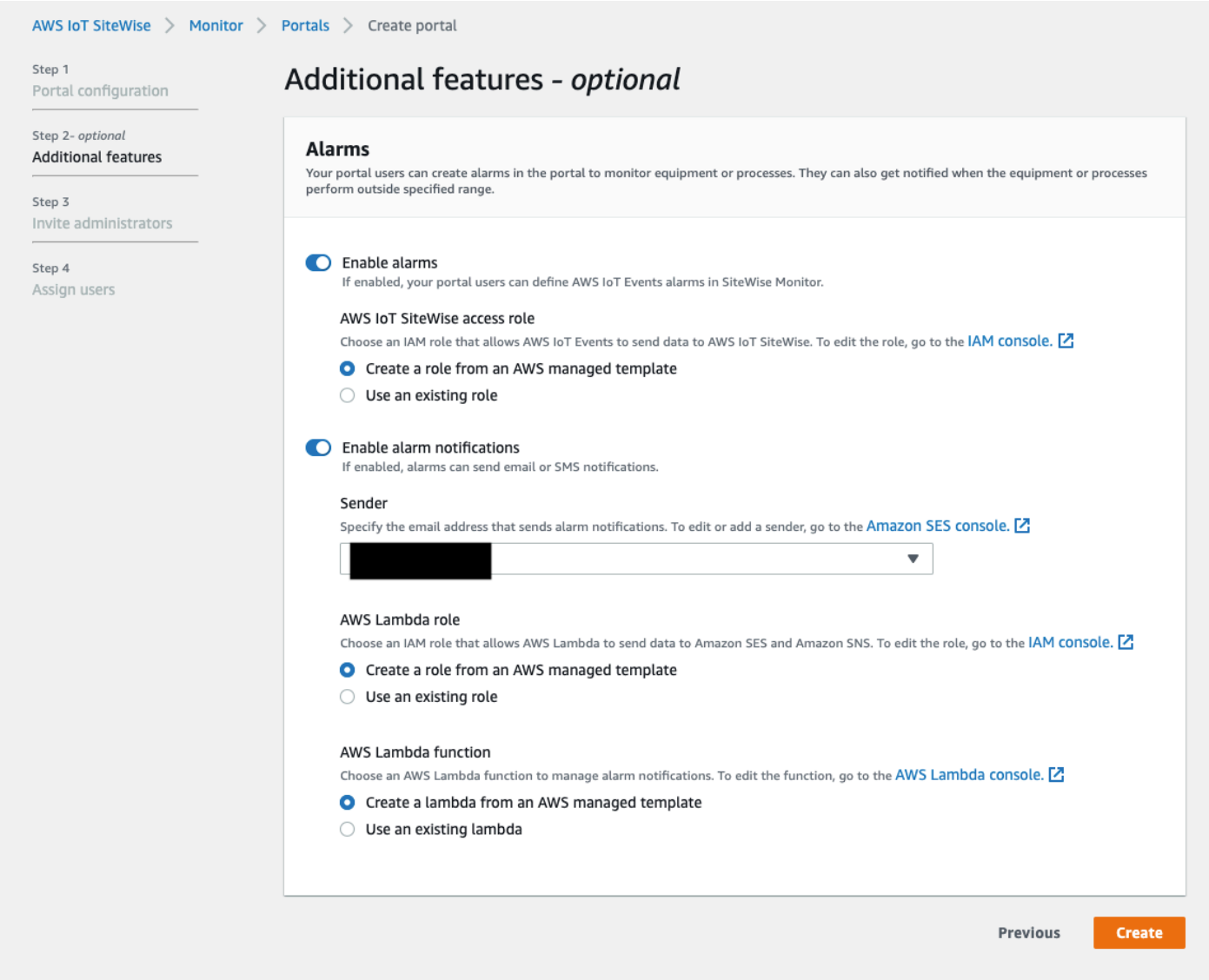
You can enable the alarms feature supported by AWS IoT Events for your portals so that portal administrators can create, edit, and delete AWS IoT Events alarm models in your SiteWise Monitor portals. Project owners can configure alarms. Project viewers can view alarm details. This section explains how you can use the AWS IoT SiteWise console to enable the alarms feature for your portals.

Important

- You can't create external alarms in your portals.
- If you want to send alarm notifications, you must choose IAM Identity Center for the user authentication service.
- The alarm notifications feature isn't available in the China (Beijing) AWS Region.

When you configure and create a portal, you can enable alarms and alarm notifications in **Step 2 Additional features**. Based on the user authentication service, choose one of the following options:

IAM Identity Center



To enable alarms for a portal

1. (Optional) Choose **Enable alarms**.

   - For **AWS IoT SiteWise access role**, use an existing role or create a role with the required permissions. This role requires the `iotevents:BatchPutMessage` permission and a trust relationship that allows `iot.amazonaws.com` and `iotevents.amazonaws.com` to assume the role.

2. (Optional) Choose **Enable alarm notifications**.

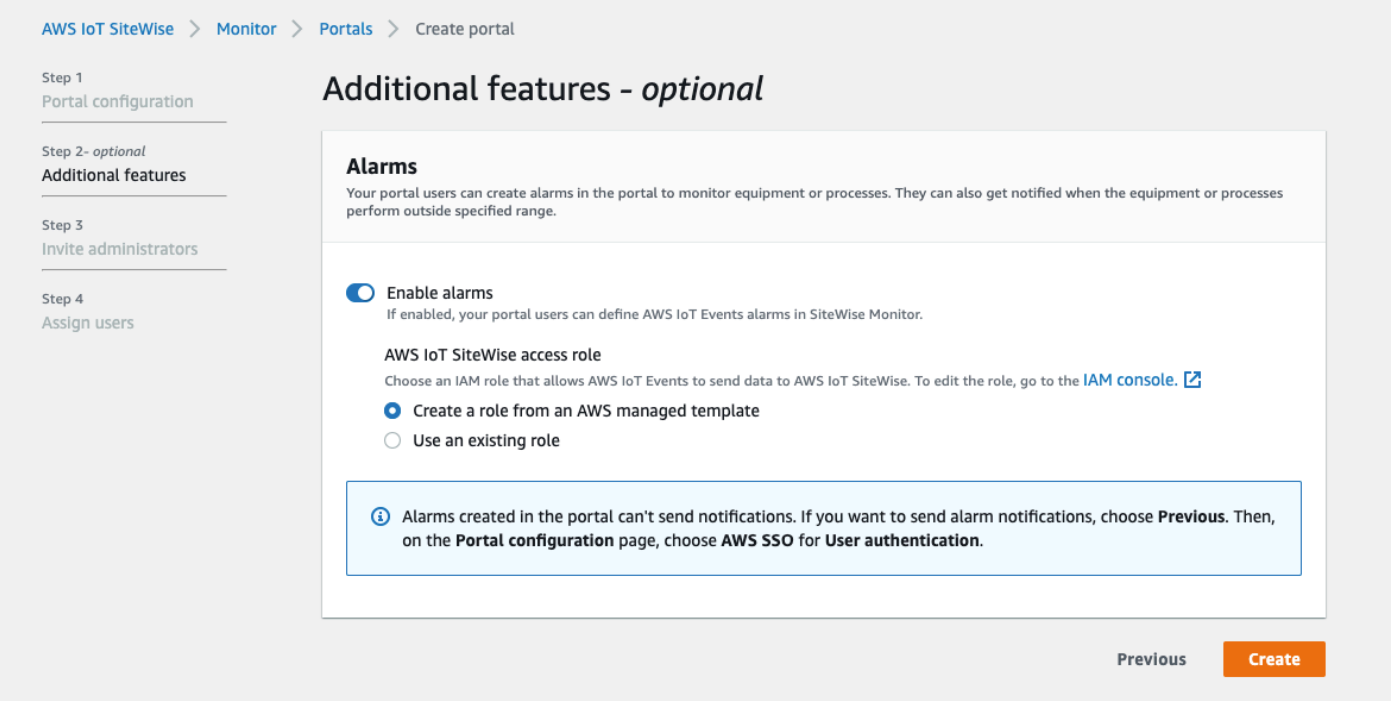
   a. For **Sender**, choose the sender.

> **Important**
>
> You must verify the sender email address in Amazon SES. For more information, see Verifying email addresses in Amazon SES, in the *Amazon Simple Email Service Developer Guide*.

b. For **AWS Lambda role**, use an existing role or create a role with the required permissions. This role requires the `lambda:InvokeFunction` and `sso-directory:DescribeUserpermissions` and a trust relationship that allows `iotevents.amazonaws.com` and `lambda.amazonaws.com` to assume the role.

c. For **AWS Lambda functions**, choose an existing Lambda function or create a function that manages alarm notifications. For more information, see Managing alarm notifications in the *AWS IoT Events Developer Guide*.

IAM



To enable alarms for a portal

- (Optional) Choose **Enable alarms**.

- For **AWS IoT SiteWise access role**, use an existing role or create a role with the required permissions. This role requires the `iotevents:BatchPutMessage` permission and a trust relationship that allows `iot.amazonaws.com` and `iotevents.amazonaws.com` to assume the role.

For more information about alarms in SiteWise Monitor, see Monitoring with alarms in the *AWS IoT SiteWise Application Guide*.

Enabling your portal at the edge

After you enable your portal at the edge, this portal is available on all SiteWise Edge gateways with the data processing pack enabled in your account.

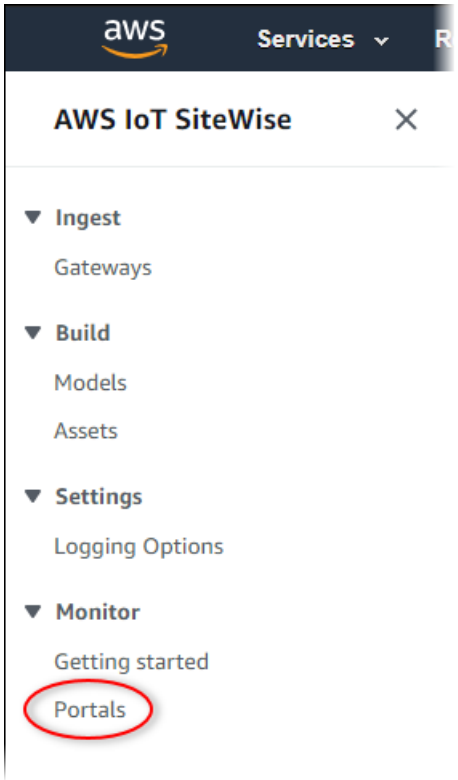
To enable the portal at the edge

1. In the **Edge configuration** section, turn on **Enable this portal at the edge**.
2. Choose **Create**.

Administering your SiteWise Monitor portals

You might need to update portal details, change administrators, or add users to your portals. This section explains how you can complete these basic administrative tasks for your SiteWise Monitor portals.

1. Sign in to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Monitor**, **Portals**.



3. Choose a portal, and then choose **View details** (or choose the portal's **Name**).
4. You can perform any of the following administrative tasks:

   - Changing a portal's name, description, branding, support email, and permissions
   - Adding or removing portal administrators
   - Sending email invitations to portal administrators
   - Adding or removing portal users
   - Deleting a portal

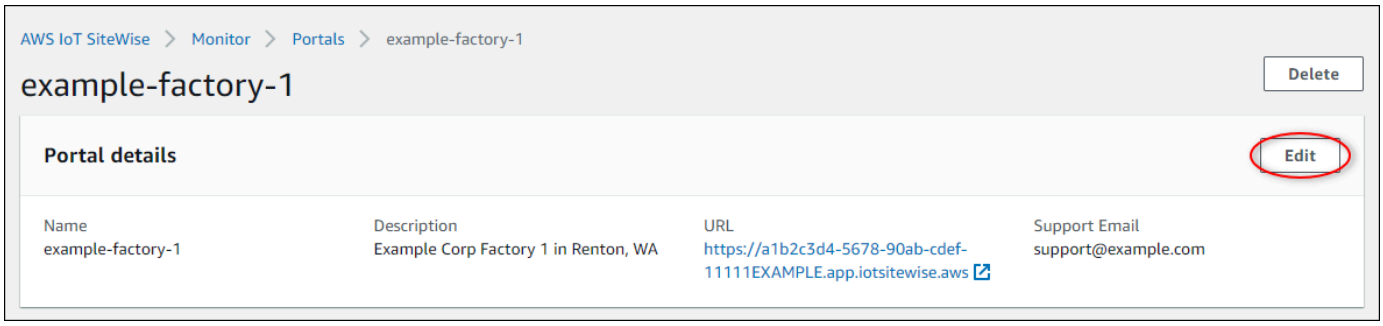
For information about how to create a portal, see Getting started with AWS IoT SiteWise Monitor.

Topics

Changing a portal's name, description, branding, support email, and permissions

You can change a portal's name, description, branding, support email, and permissions.

1. On the portal details page, in the **Portal details** section, choose **Edit**.

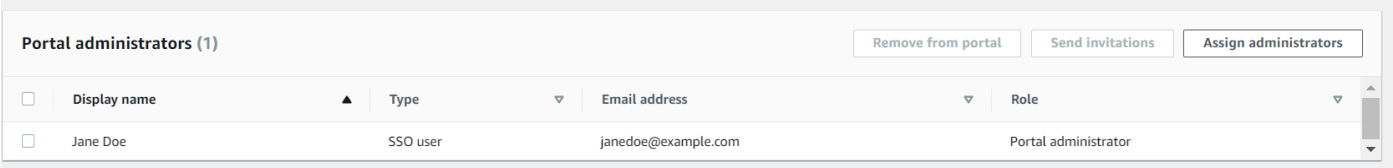


2. Update the **Name**, **Description**, **Portal branding**, **Support contact email**, or **Permissions**.
3. When you're finished, choose **Save**.

Adding or removing portal administrators

In a few steps, you can add or remove users as administrators for a portal. Based on the user authentication service, choose one of the following options.

IAM Identity Center

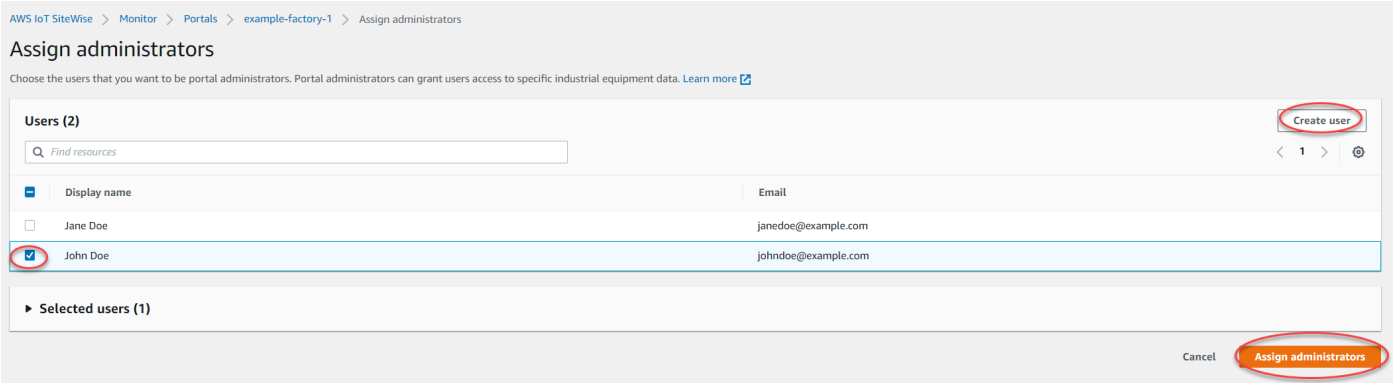


To add portal administrators

1. On the portal details page, in the **Portal administrators** section, choose **Assign administrators**.
2. On the **Assign administrators** page, select the check boxes for the users to add to the portal as administrators.

> **Note**
>
> If you use IAM Identity Center as your identity store, and you're signed in to your AWS Organizations management account, you can choose **Create user** to create an IAM Identity Center user. IAM Identity Center sends the new user an email for them to set their password. You can then assign the user to the portal as an administrator. For more information, see Manage identities in IAM Identity Center.

3. Choose **Assign administrators**.

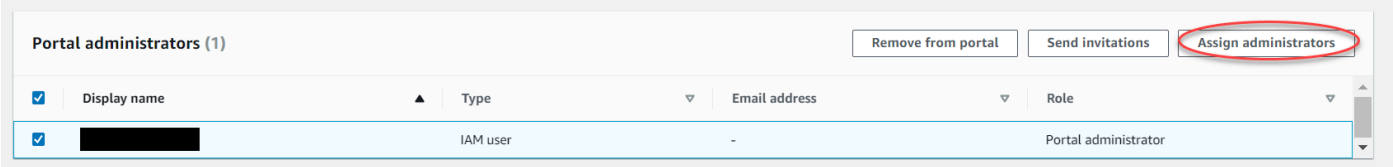


To remove portal administrators

- On the portal details page, in the **Portal administrators** section, select the check box for each user to remove, and then choose **Remove from portal**.

> **Note**
>
> We recommend that you select at least one portal administrator.

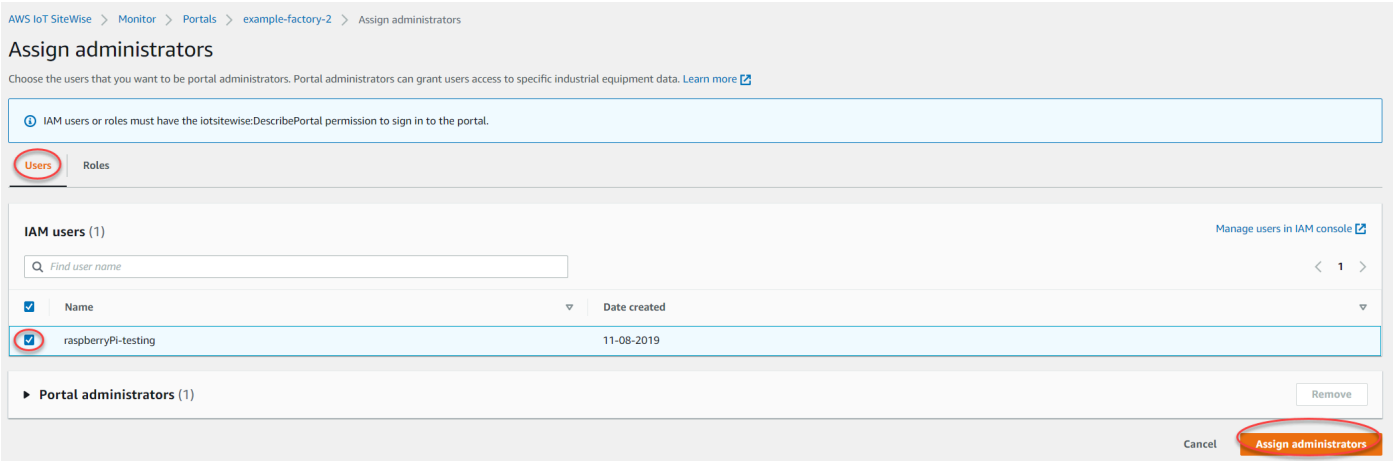
IAM

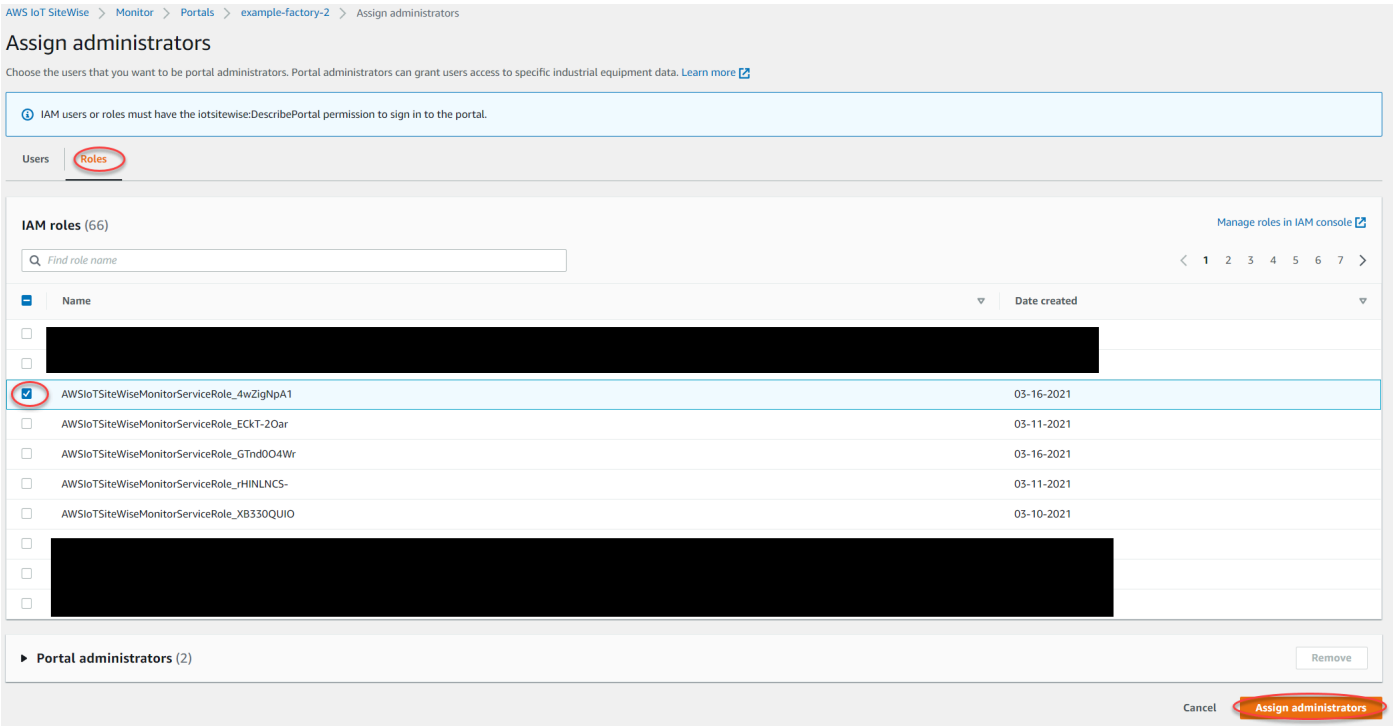


To add portal administrators

1. On the portal details page, in the **Portal administrators** section, choose **Assign administrators**.
2. On the **Assign administrators** page, do the following:
   - Choose **IAM users**, if you want to add an IAM user as your portal administrator.
   - Choose **IAM roles**, if you want to add an IAM role as your portal administrator.
3. Select the check boxes for the users or roles that you want as your portal administrators. This adds the users or roles to the **Portal administrators** list.
4. Choose **Assign administrators**.

> **Important**
>
> Users or roles must have the `iotsitewise:DescribePortal` permission to sign in to the portal.





To remove portal administrators

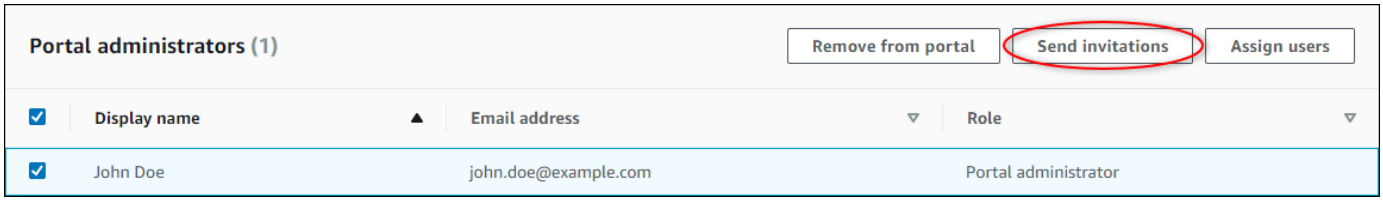
- On the portal details page, in the **Portal administrators** section, select the check box for each user to remove, and then choose **Remove from portal**.

> **Note**
>
> Leaving a portal without a portal administrator is not recommended.

Sending email invitations to portal administrators

You can send email invitations to portal administrators.

1. On the portal details page, in the **Portal administrators** section, select the check boxes for the portal administrators.



2. Choose **Send invitations**. Your email client opens, and an invitation is populated in the message body.

   You can customize the email before you send it to your portal administrators.

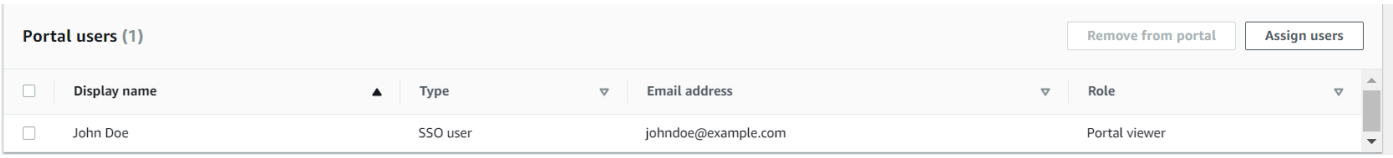
Adding or removing portal users

You choose which users have access to your portals. Portal users appear in the list of users within a SiteWise Monitor portal. From this list, portal administrators can add project owners, and project owners can add project viewers.

> **Note**
>
> Your portal administrators and portal users might contact you through a portal's support email if they need you to add or remove a user.

Based on the user authentication service, choose one of the following options.

IAM Identity Center

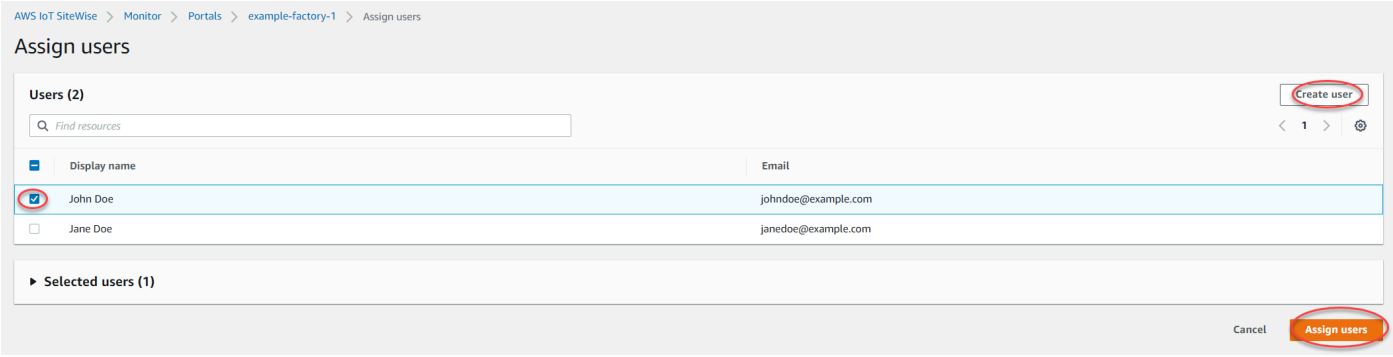


To add portal users

1. On the portal details page, in the **Portal users** section, choose **Assign users**.
2. On the **Assign users** page, select the check box for the users to add to the portal.

   > **Note**
   >
   > If you use IAM Identity Center as your identity store, and you're signed in to your AWS Organizations management account, you can choose **Create user** to create an IAM Identity Center user. IAM Identity Center sends the new user an email for them to set their password. You can then assign the user to the portal as a user. For more information, see Manage identities in IAM Identity Center.

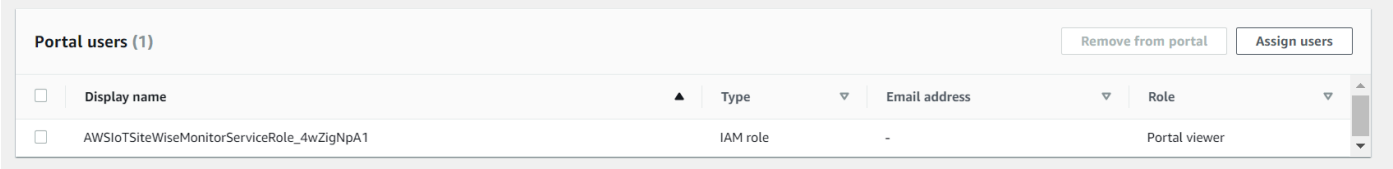
3. Choose **Assign users**.



To remove portal users

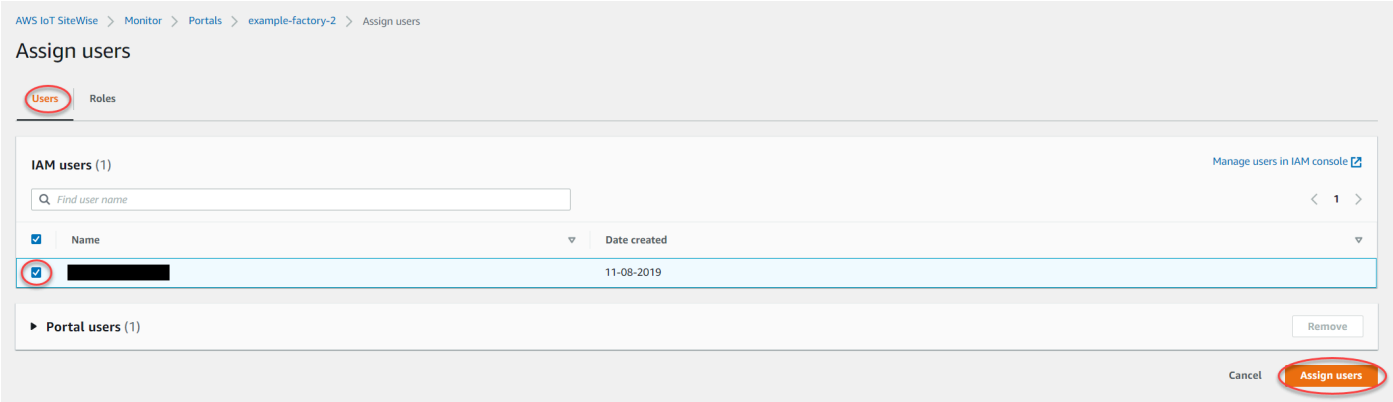
- On the portal details page, in the **Portal users** section, select the check box for the users to remove from the portal, and then choose **Remove from portal**.

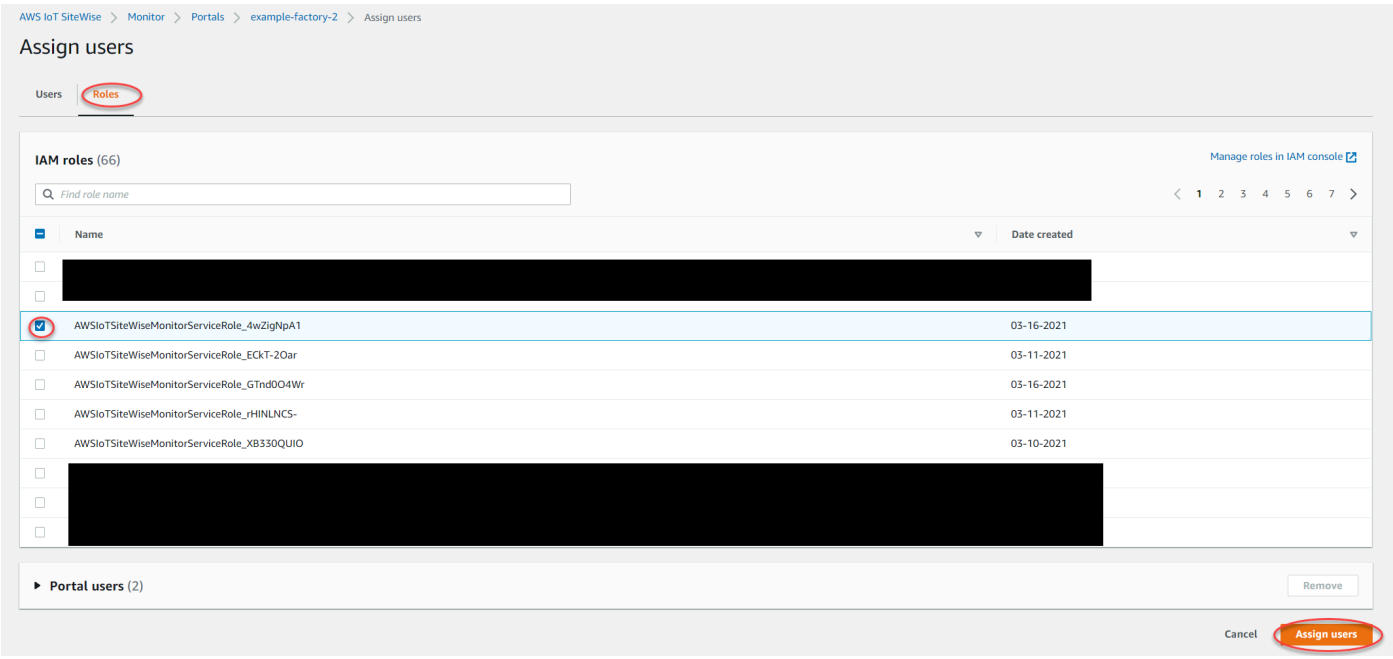
IAM



To add portal users

1. On the portal details page, in the **Portal users** section, choose **Assign users**.
2. On the **Assign users** page, do the following:
   - Choose **IAM users** to add an IAM user as your portal user.
   - Choose **IAM roles** to add an IAM role as your portal user.
3. Select the check boxes for the users or roles that you want to add as your portal users. This adds the users or roles to the **Portal users** list.
4. Choose **Assign users**.





To remove portal users

- On the portal details page, in the **Portal users** section, select the check box for the users to remove from the portal, and then choose **Remove from portal**.

> **Important**
>
> Users or roles must have the `iotsitewise:DescribePortal` permission to sign in to the portal.

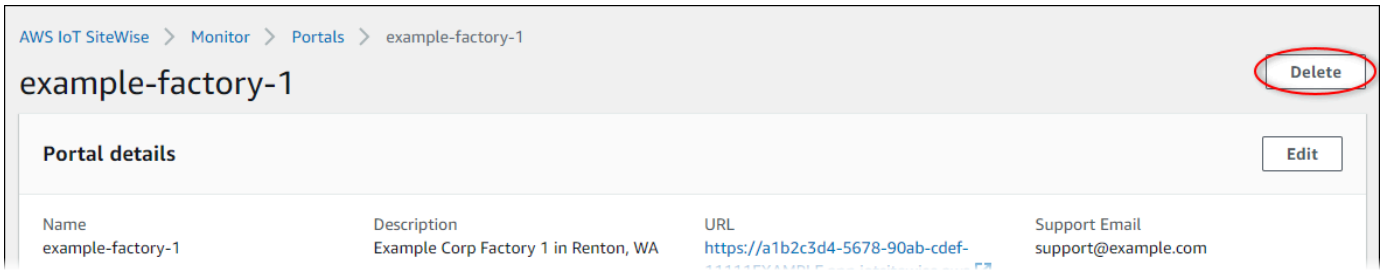
Deleting a portal

You might delete a portal if you created it for testing purposes or if you created a duplicate of a portal that already exists.

> **Note**
>
> You must first manually delete all dashboards and projects in a portal before you can delete a portal. For more information, see Deleting projects and Deleting dashboards in the *SiteWise Monitor Application Guide*.

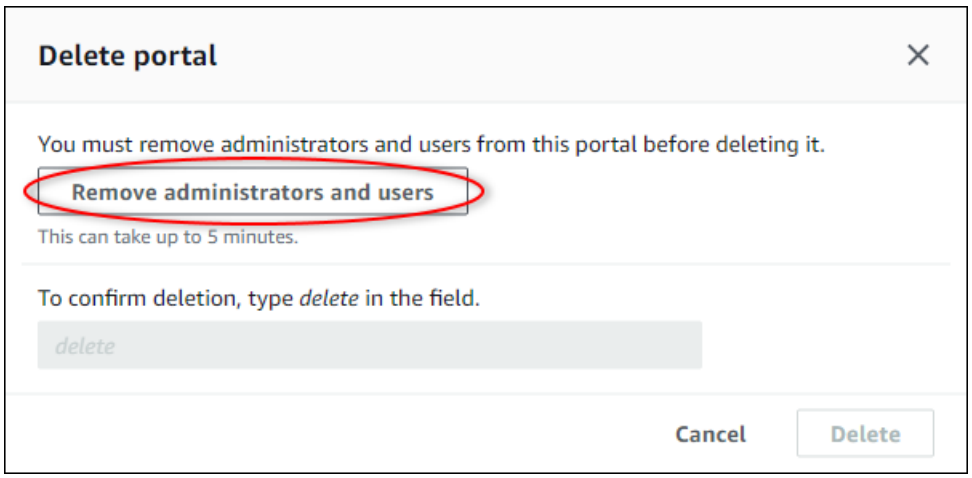
1. On the portal details page, choose **Delete**.

   > **Important**
   >
   > When you delete a portal, you lose all projects that the portal contains, and all dashboards in each project. This action can't be undone. Your asset data isn't affected.

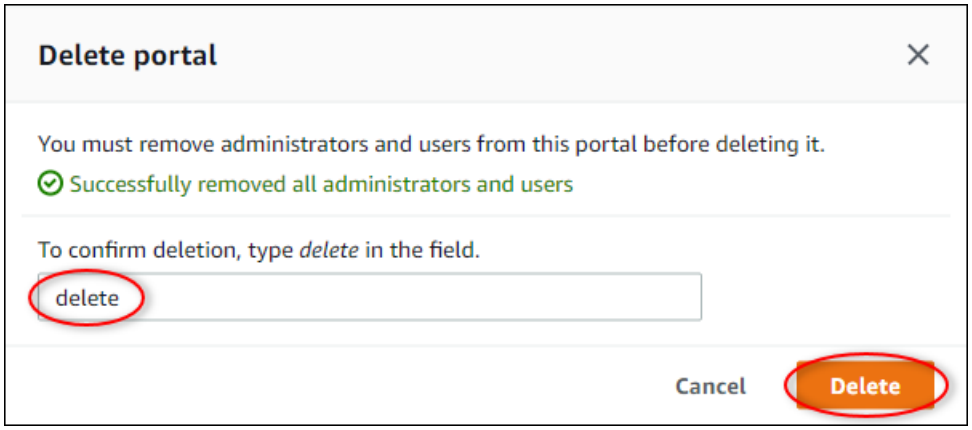


2. In the **Delete portals** dialog box, choose **Remove admins and users**.

   You must remove the administrators and users from a portal before you can delete it. If your portal doesn't have administrators or users, the button doesn't appear, and you can skip to the next step.



3. If you're sure that you want to delete the entire portal, enter **`delete`** in the field to confirm deletion.



4. Choose **Delete**.

Monitoring data with IoT dashboard application

IoT dashboard application is an open source dashboard application where you can visualize and interact with operational data. You can utilize the AWS Cloud Development Kit (AWS CDK) to deploy IoT dashboard application.

The following are examples of the customizable data visualization features in IoT dashboard application:

- Support for multiple properties in a single line chart.
- Enhanced search of assets and properties.

Customers from manufacturing, logistics, energy, and other industries can use IoT dashboard application to address specific challenges like tracking equipment performance, optimizing operational efficiency, and data-driven decisions. For more information, see GitHub repository for IoT dashboard application.

Query data from AWS IoT SiteWise

You can use the AWS IoT SiteWise API operations to query your asset properties' current values, historical values, and aggregates over specific time intervals.

Use these features to gain insight into your data. For example, discover all your assets with a given property value or build a custom representation of your data. You can also use API operations to develop software solutions that integrate with the industrial data stored in your AWS IoT SiteWise assets. You can also explore your asset data live in AWS IoT SiteWise Monitor. To learn how to configure SiteWise Monitor, see Monitoring data with AWS IoT SiteWise Monitor .

The operations described in this section return property value objects that contain timestamp, quality, value (TQV) structures:

- The `timestamp` contains the current Unix epoch time in seconds with nanosecond offset.
- The `quality` contains one of the following strings that indicate the quality of the data point:
  - `GOOD` – The data isn't affected by any issues.
  - `BAD` – The data is affected by an issue such as sensor failure.
  - `UNCERTAIN` – The data is affected by an issue such as sensor inaccuracy.
- The `value` contains one of the following fields, depending on the type of the property:
  - `booleanValue`
  - `doubleValue`
  - `integerValue`
  - `stringValue`

Topics

Querying current asset's property values

This tutorial shows two ways to get the current value of an asset property. You can use the AWS IoT SiteWise console or use API in the AWS Command Line Interface (AWS CLI).

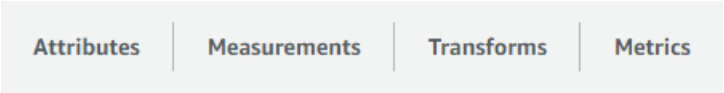
Topics

Query an asset property's current value (console)

You can use the AWS IoT SiteWise console to view the current value of an asset property.

To get the current value of an asset property (console)

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**.
3. Choose the asset with the property to query.
4. Choose the arrow icon to expand an asset hierarchy to find your asset.
5. Choose the tab for the type of property. For example, choose **Measurements** to view the current value of a measurement property.



6. Find the property to view. The current value appears in the **Latest value** column.

Query an asset property's current value (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to query the current value of an asset property.

Use the GetAssetPropertyValue operation to query an asset property's current value.

To identify an asset property, specify one of the following:

- The `assetId` and `propertyId` of the asset property that data is sent to.

- The `propertyAlias`, which is a data stream alias (for example, `/company/windfarm/3/turbine/7/temperature`). To use this option, you must first set your asset property's alias. To set property aliases, see Mapping industrial data streams to asset properties.

To get the current value of an asset property (AWS CLI)

- Run the following command to get the current value of the asset property. Replace *`asset-id`* with the ID of the asset and *`property-id`* with the ID of the property.

```
aws iotsitewise get-asset-property-value \
  --asset-id asset-id \
  --property-id property-id
```

The operation returns a response that contains the current TQV of the property in the following format.

```
{
  "propertyValue": {
    "value": {
      "booleanValue": Boolean,
      "doubleValue": Number,
      "integerValue": Number,
      "stringValue": "String"
    },
    "timestamp": {
      "timeInSeconds": Number,
      "offsetInNanos": Number
    },
    "quality": "String"
  }
}
```

Querying historical asset property values

You can use the AWS IoT SiteWise API GetAssetPropertyValueHistory operation to query the historical values of an asset property.

To identify an asset property, specify one of the following:

- The `assetId` and `propertyId` of the asset property that data is sent to.
- The `propertyAlias`, which is a data stream alias (for example, `/company/windfarm/3/turbine/7/temperature`). To use this option, you must first set your asset property's alias. To set property aliases, see Mapping industrial data streams to asset properties.

Pass the following parameters to refine your results:

- `startDate` – The exclusive start of the range from which to query historical data, expressed in seconds in Unix epoch time.
- `endDate` – The inclusive end of the range from which to query historical data, expressed in seconds in Unix epoch time.
- `maxResults` – The maximum number of results to return in one request. Defaults to `20` results.
- `nextToken` – A pagination token returned from a previous call of this operation.
- `timeOrdering` – The ordering to apply to the returned values: `ASCENDING` or `DESCENDING`.
- `qualities` – The quality to filter results by: `GOOD`, `BAD`, or `UNCERTAIN`.

Topics

Query the value history for an asset property (AWS CLI)

To query the value history for an asset property (AWS CLI)

1. Run the following command to get the value history for the asset property. This command queries the property's history over a specific 10 minute interval. Replace *`asset-id`* with the ID of the asset and *`property-id`* with the ID of the property. Replace the date parameters with the interval to query.

```
aws iotsitewise get-asset-property-value-history \
  --asset-id asset-id \
  --property-id property-id \
  --start-date 1575216000 \
  --end-date 1575216600
```

The operation returns a response that contains the historical TQVs of the property in the following format:

```
{
  "assetPropertyValueHistory": [
    {
      "value": {
        "booleanValue": Boolean,
        "doubleValue": Number,
        "integerValue": Number,
        "stringValue": "String"
      },
      "timestamp": {
        "timeInSeconds": Number,
        "offsetInNanos": Number
      },
      "quality": "String"
    }
  ],
  "nextToken": "String"
}
```

2. If more value entries exist, you can pass the pagination token from the `nextToken` field to a subsequent call to the GetAssetPropertyValueHistory operation.

Querying asset property aggregates

AWS IoT SiteWise automatically computes aggregated asset property values, which are a set of basic metrics calculated over multiple time intervals. AWS IoT SiteWise computes the following aggregates every minute, hour, and day for your asset properties:

- **average** – The average (mean) of a property's values over a time interval.
- **count** – The number of data points for a property over a time interval.
- **maximum** – The maximum of a property's values over a time interval.
- **minimum** – The minimum of a property's values over a time interval.
- **standard deviation** – The standard deviation of a property's values over a time interval.
- **sum** – The sum of a property's values over a time interval.

For non-numeric properties, such as strings and Booleans, AWS IoT SiteWise computes only the count aggregate.

You can also compute custom metrics for your asset data. With metric properties, you define aggregations that are specific to your operation. Metric properties offer additional aggregation functions and time intervals that aren't precomputed for the AWS IoT SiteWise API. For more information, see Aggregating data from properties and other assets (metrics).

Topics

Aggregates for an asset property (API)

You can use the AWS IoT SiteWise API to get aggregates for an asset property.

Use the GetAssetPropertyAggregates operation to query aggregates of an asset property.

To identify an asset property, specify one of the following:

- The `assetId` and `propertyId` of the asset property that data is sent to.
- The `propertyAlias`, which is a data stream alias (for example, `/company/windfarm/3/turbine/7/temperature`). To use this option, you must first set your asset property's alias. To set property aliases, see Mapping industrial data streams to asset properties.

You must also pass the following required parameters:

- `aggregateTypes` – The list of aggregates to retrieve. You can specify any of `AVERAGE`, `COUNT`, `MAXIMUM`, `MINIMUM`, `STANDARD_DEVIATION`, and `SUM`.
- `resolution` – The time interval for which to retrieve the metric: `1m` (1 minute), `1h` (1 hour), or `1d` (1 day).
- `startDate` – The exclusive start of the range from which to query historical data, expressed in seconds in Unix epoch time.
- `endDate` – The inclusive end of the range from which to query historical data, expressed in seconds in Unix epoch time.

You can also pass any of the following parameters to refine your results:

- `maxResults` – The maximum number of results to return in one request. Defaults to `20` results.
- `nextToken` – A pagination token returned from a previous call of this operation.
- `timeOrdering` – The ordering to apply to the returned values: `ASCENDING` or `DESCENDING`.
- `qualities` – The quality to filter results by: `GOOD`, `BAD`, or `UNCERTAIN`.

> **Note**
>
> The GetAssetPropertyAggregates operation returns a TQV with a different format than other operations described in this section. The `value` structure contains a field for each of the `aggregateTypes` in the request. The `timestamp` contains the time that the aggregation occurred, in seconds in Unix epoch time.

Aggregates for an asset property (AWS CLI)

To query aggregates for an asset property (AWS CLI)

1. Run the following command to get aggregates for the asset property. This command queries the average and sum with a 1 hour resolution for a specific 1 hour interval. Replace *`asset-id`* with the ID of the asset and *`property-id`* with the ID of the property. Replace the parameters with the aggregates and interval to query.

```
aws iotsitewise get-asset-property-aggregates \
  --asset-id asset-id \
  --property-id property-id \
  --start-date 1575216000 \
  --end-date 1575219600 \
  --aggregate-types AVERAGE SUM \
  --resolution 1h
```

   The operation returns a response that contains the historical TQVs of the property in the following format. The response includes only the requested aggregates.

```
{
  "aggregatedValues": [
    {
      "timestamp": Number,
      "quality": "String",
```

```
      "value": {
        "average": Number,
        "count": Number,
        "maximum": Number,
        "minimum": Number,
        "standardDeviation": Number,
        "sum": Number
      }
    }
  ],
  "nextToken": "String"
}
```

2. If more value entries exist, you can pass the pagination token from the `nextToken` field to a subsequent call to the GetAssetPropertyAggregates operation.

AWS IoT SiteWise query language

With the AWS IoT SiteWise data retrieval ExecuteQuery API operation, you can retrieve information about declarative structural definitions, and the timeseries data associated with them, from the following:

- models
- assets
- measurements
- metrics
- transforms
- aggregates

This can be done with SQL like query statements, in a single API request.

> **Note**
>
> This feature is available in all Regions where both AWS IoT SiteWise and AWS IoT TwinMaker are available, except AWS GovCloud (US-West).

Topics

Prerequisites

AWS IoT SiteWise requires permissions to integrate with AWS IoT TwinMaker so that it can organize and model industrial data.

Before you can retrieve information about models, assets, measurements, metrics, transforms, and aggregates, ensure the following prerequisites are met:

- Service-linked roles for both AWS IoT SiteWise and AWS IoT TwinMaker setup in your AWS account. For more information about service-linked roles, see Using service-linked roles in the *IAM User Guide*.
- An enabled AWS IoT SiteWise integration for your IAM role. For more information, see Integrating AWS IoT SiteWise and AWS IoT TwinMaker.
- An AWS IoT TwinMaker workspace with ID `IoTSiteWiseDefaultWorkspace` in your account in the Region. For more information, see Using the IoTSiteWiseDefaultWorkspace in the *AWS IoT TwinMaker User Guide*.
- Either the **standard** or **tiered bundle** pricing modes for AWS IoT TwinMaker enabled. For more information, see Switch AWS IoT TwinMaker pricing modes in the *AWS IoT TwinMaker User Guide*.

Query language reference

AWS IoT SiteWise supports a rich query language for working with your data. The available data types, operators, functions and constructs are described in the following topics.

See Example queries to write queries with the AWS IoT SiteWise query language.

Topics

- Example queries

Understanding views

This section provides information to help you understand the views in AWS IoT SiteWise, such as process metadata and telemetry data.

The following tables provide the view names and descriptions of the views.

Data model

View name	View description
asset	Contains information about the asset and model derivation.
asset_property	Contains information about the asset property's structure.
raw_time_series	Contains the historical data of the time series.
latest_value_time_series	Contains the latest value of the time series.
precomputed_aggregates	Contains the automatically computed aggregated asset property values. They are a set of basic metrics calculated over multiple time intervals.

The following views list the column names for queries along with sample data.

View:asset

asset_id	asset_name	asset_description	asset_model_id
88898498-0b8b-42b5-bf57-16180bc3d3a0	WindTurbine A	WindTurbine Asset A	17847250-5bf0-4f74-b775-cc03f05e7cb8
17847250-5bf0-4f74-b775-cc03f05e7cb8	Wind Turbine Asset Model	Represents a turbine in a wind farm.	

View:asset_property

property_id	asset_id	property_name	property_data_type	property_alias	asset_composite_model_id
b29be434-b000-4d74-b809-75287d83bcd6	88898498-0b8b-42b5-bf57-16180bc3d3a0	motor temperature	Double	Rochester 2/44///Line-5/Bus-2/Machine-5/Temperature	
3b458f00-24e7-458a-b4e8-c6026eff654a	88898498-0b8b-42b5-bf57-16180bc3d3a0	wind direction	Double	/company/windfarm/3/turbine/7/winddirection	2f458n00-56e7-458h-b4e8-c6026eff985g

View:raw_time_series

asset_id	property_id	property_alias	event_timestamp	quality	boolean_value	int_value	double_value	string_value
88898498-0b8b-42b-bf57-16180bc3d3a	b29be434-b000-4d7-b809-75287d83bcd	Rochest 2/44/// Line-5/ Bus-2/ Machine-5/ Temperature	1575219 0	GOOD			115.0	
88898498-0b8b-42b-	3b458f00-24e7-458-b4e8-	/ company/	1575219 7	GOOD			348.75	

asset_id	property_id	property_alias	event_timestamp	quality	boolean_value	int_value	double_value	string_value
bf57-16180bc3d3a	c6026eff654a	windfarm/3/turbine/7/winddirection						

Note

You must include a filter clause on the `event_timestamp` column to query the `raw_time_series` view. This is a required filter, and the query will fail without it.

Example query

```
SELECT event_timestamp, double_value FROM raw_time_series WHERE event_timestamp
 > 1234567890
```

View:latest_value_time_series

asset_id	property_id	property_alias	event_timestamp	quality	boolean_value	int_value	double_value	string_value
88898498-0b8b-42b-bf57-16180bc3d3a	3b458f00-24e7-458-b4e8-c6026eff654a	/company/windfarm/3/turbine/7/winddirection	1575219600	GOOD			355.39	

View:precomputed_aggregates

asset_id	property id	property alias	event_timestamp	resolution	sum_value	count_value	average value	maximum value	minimum value	stdev_value
888984 0b8b-4 - bf57-1 80bc3d	b29be4 b000-4 - b809-7 87d83b	Roches 2/44// Line-5/ Bus- 2/ Machin -5/ Temper ature	157521 0	15m	1105.4	15	73.4	80.6	68	3.64

Supported data types

AWS IoT SiteWise query language supports the following data types.

View:asset

Data type	Description
`STRING`	A string of maximum length 1024 bytes.
`INTEGER`	A signed 32-bit integer with a range from `-2,147,483,648 to 2,147,483,647` .
`DOUBLE`	A floating point number with range from – `10^100 to 10^100` and `IEEE 754` double precision.
`BOOLEAN`	`true` or `false`.

> **Note**
>
> The double precision data is not exact. Some values are not converted exactly, and will not represent all real numbers due to limited precision. Floating-point data in the query may not be the same value represented internally. The value is rounded if the precision of an input number is too high.

Retrieve data with a SELECT statement

The SELECT statement is used to retrieve data from one or more views. AWS IoT SiteWise supports an implicit JOIN of the views. You can list the views to join (in the FROM clause of the SELECT statement), using commas to separate them.

Example

Use the following SELECT statement:

```
SELECT select_expr [, ...]
[ FROM from_item [, ...] ]
[ WHERE [LIKE condition ESCAPE condition] ]
```

In the previous example, the LIKE clause specifies the search and filtering conditions using wild cards. AWS IoT SiteWise supports `percentage (%)` as the wild card character.

Example to use % in a condition:

```
Prefix search: String%
Infix search: %String%
Suffix search: %String
```

Example to search for an asset:

```
SELECT asset_name, asset_description FROM asset WHERE asset_name LIKE 'Wind%'
```

Example to search for an asset using an ESCAPE condition:

```
SELECT asset_name, asset_description FROM asset WHERE asset_name LIKE 'room\%' ESCAPE
 '\'
```

Logical operators

AWS IoT SiteWise supports the following logical operators.

Logical operators

Operator	Description	Example
AND	TRUE if both values are true	a AND b

If either a or b is FALSE, the previous expression evaluates to false. For an AND operator to evaluate to true, both a and b must be true.

Example

```
SELECT a.asset_name
FROM asset as a, latest_value_time_series as t
WHERE t.int_value > 30 AND t.event_timestamp > 1234567890
```

Comparison operators

AWS IoT SiteWise supports the following comparison operators.

Logical operators

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
=	Equals
!=	Not equal

Example queries

Metadata filtering

The following example is for metadata filtering with a SELECT statement with the AWS IoT SiteWise query language:

```
SELECT a.asset_name, p.property_name
FROM asset a, asset_property p
WHERE a.asset_id = p.asset_id AND a.asset_name LIKE '%windmill%'
```

Value filtering

The following is an example of value filtering using a SELECT statement with the AWS IoT SiteWise query language:

```
SELECT a.asset_name FROM asset a, raw_time_series r
WHERE a.asset_id = r.asset_id AND r.int_value > 30  AND r.event_timestamp > 1234567890
 AND r.event_timestamp < 1234567891
```

Interacting with other AWS services

AWS IoT SiteWise can publish asset data to the AWS IoT MQTT publish-subscribe message broker, so that you can interact with your asset data from other AWS services. AWS IoT SiteWise assigns each asset property a unique MQTT topic that you can use to route your asset data to other AWS services using AWS IoT Core rules. For example, you can configure AWS IoT Core rules to do the following tasks:

- Identify equipment failure and notify appropriate personnel by sending data to AWS IoT Events.
- Historize select asset data for use in external software solutions by sending data to Amazon DynamoDB.
- Generate weekly reports by triggering an AWS Lambda function.

You can follow a tutorial that walks through the steps required to set up a rule that stores property values in DynamoDB. For more information, see Publishing property value updates to Amazon DynamoDB.

For more information about how to configure a rule, see Rules in the *AWS IoT Developer Guide*.

You can also consume data from other AWS services back into AWS IoT SiteWise. To ingest data through the AWS IoT SiteWise rule action, see Ingesting data using AWS IoT Core rules.

Topics

Understanding asset properties' MQTT topics

Every asset property has a unique MQTT topic path in the following format.

```
$aws/sitewise/asset-models/assetModelId/assets/assetId/properties/propertyId
```

Note

AWS IoT SiteWise doesn't support the # (multi-level) topic filter wildcard in the AWS IoT Core rules engine. You can use the + (single-level) wildcard. For example, you can use the following topic filter to match all updates for a particular asset model.

```
$aws/sitewise/asset-models/assetModelId/assets/+/properties/+
```

To learn more about topic filter wildcards, see Topics in the *AWS IoT Core Developer Guide*.

Working with asset property notifications

You can enable property notifications to publish asset data updates to AWS IoT Core, and then run queries on the your data. With asset property notifications, AWS IoT SiteWise provides an AWS CloudFormation template that you can use to export AWS IoT SiteWise data to Amazon S3.

Note

Asset data is sent to AWS IoT Core every time it's received by AWS IoT SiteWise, regardless of if the value has changed.

Topics

Enabling asset property notifications (console)

By default, AWS IoT SiteWise doesn't publish property value updates. You can use the AWS IoT SiteWise console to enable notifications for an asset property.

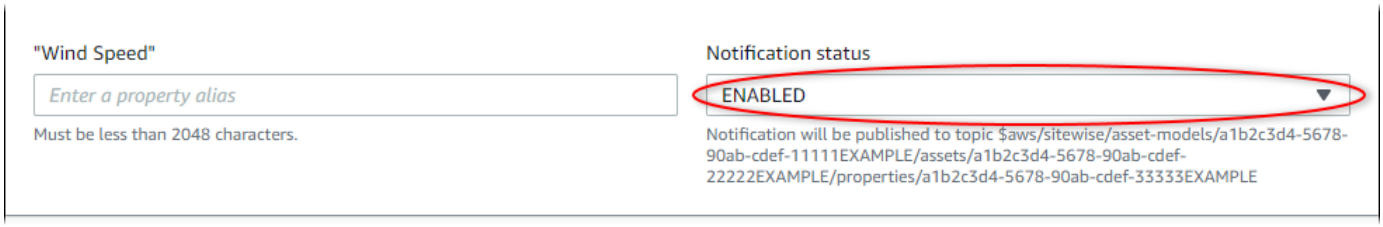
To enable or disable notifications for an asset property (console)

1. Navigate to the AWS IoT SiteWise console.

2. In the navigation pane, choose **Assets**.
3. Choose the asset to enable a property's notifications.

> **Tip**
>
> You can choose the arrow icon to expand an asset hierarchy to find your asset.

4. Choose **Edit**.
5. For the asset property's **Notification status**, choose **ENABLED**.



   You can also choose **DISABLED** to disable notifications for the asset property.
6. Choose **Save**.

Enabling asset property notifications (AWS CLI)

By default, AWS IoT SiteWise doesn't publish property value updates. You can use the AWS Command Line Interface (AWS CLI) to enable or disable notifications for an asset property.

You must know your asset's `assetId` and property's `propertyId` to complete this procedure. You can also use the external ID. If you created an asset and don't know its `assetId`, use the ListAssets API to list all the assets for a specific model. Use the DescribeAsset operation to view your asset's properties including property IDs.

Use the UpdateAssetProperty operation to enable or disable notifications for an asset property. Specify the following parameters:

- `assetId` – The asset's ID.
- `propertyId` – The asset property's ID.
- `propertyNotificationState` – The property value notification state: `ENABLED` or `DISABLED`.
- `propertyAlias` – The alias of the property. Specify the property's existing alias when you update the notification state. If you omit this parameter, the property's existing alias is removed.

To enable or disable notifications for an asset property (CLI)

1. Run the following command to retrieve the asset property's alias. Replace *asset-id* with the ID of the asset and *property-id* with the ID of the property.

```
aws iotsitewise describe-asset-property \
  --asset-id asset-id \
  --property-id property-id
```

   The operation returns a response that contains the asset property's details in the following format. The property alias is in `assetProperty.alias` in the JSON object.

```
{
  "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
  "assetName": "Wind Turbine 7",
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
  "assetProperty": {
    "id": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
    "name": "Wind Speed",
    "alias": "/company/windfarm/3/turbine/7/windspeed",
    "notification": {
      "topic": "$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/
assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE",
      "state": "DISABLED"
    },
    "dataType": "DOUBLE",
    "unit": "m/s",
    "type": {
      "measurement": {}
    }
  }
}
```

2. Run the following command to enable notifications for the asset property. Replace *property-alias* with the property alias from the previous command's response, or omit `--property-alias` to update the property without an alias.

```
aws iotsitewise update-asset-property \
  --asset-id asset-id \
  --property-id property-id \
  --property-notification-state ENABLED \
```

```
  --property-alias property-alias
```

You can also pass `--property-notification-state DISABLED` to disable notifications for the asset property.

Querying asset property notification messages

To query asset property notifications, create AWS IoT Core rules made up of SQL statements.

AWS IoT SiteWise publishes asset property data updates to AWS IoT Core in the following format.

```
{
  "type": "PropertyValueUpdate",
  "payload": {
    "assetId": "String",
    "propertyId": "String",
    "values": [
      {
        "timestamp": {
          "timeInSeconds": Number,
          "offsetInNanos": Number
        },
        "quality": "String",
        "value": {
          "booleanValue": Boolean,
          "doubleValue": Number,
          "integerValue": Number,
          "stringValue": "String"
        }
      }
    ]
  }
}
```

Each structure in the `values` list is a timestamp-quality-value (TQV) structure.

- The `timestamp` contains the current Unix epoch time in seconds with nanosecond offset.
- The `quality` contains one of the following strings that indicate the quality of the data point:
  - GOOD – The data isn't affected by any issues.
  - BAD – The data is affected by an issue such as sensor failure.

  - `UNCERTAIN` – The data is affected by an issue such as sensor inaccuracy.
- The `value` contains one of the following fields, depending on the type of the property:
  - `booleanValue`
  - `doubleValue`
  - `integerValue`
  - `stringValue`

To parse values out of the `values` array, you need to use complex nested object queries in your rules' SQL statements. For more information, see Nested object queries in the *AWS IoT Developer Guide*, or see the Publishing property value updates to Amazon DynamoDB tutorial for a specific example of parsing asset property notification messages.

Example Example query to extract the array of values

The following statement demonstrates how to query the array of updated property values for a specific double-type property on all assets with that property.

```
SELECT
  (SELECT VALUE (value.doubleValue) FROM payload.values) AS windspeed
FROM
  '$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/assets/+/
properties/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE'
WHERE
  type = 'PropertyValueUpdate'
```

The previous rule query statement outputs data in the following format.

```
{
  "windspeed": [
    26.32020195042838,
    26.282584572975477,
    26.352566977372508,
    26.283084346171442,
    26.571883739599322,
    26.60684140743005,
    26.628738636715045,
    26.273486932802125,
    26.436379105473964,
    26.600590095377303
```

```
  ]
}
```

Example Example query to extract a single value

The following statement demonstrates how to query the first value from the array of property values for a specific double-type property on all assets with that property.

```
SELECT
  get((SELECT VALUE (value.doubleValue) FROM payload.values), 0) AS windspeed
FROM
  '$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/assets/+/
properties/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE'
WHERE
  type = 'PropertyValueUpdate'
```

The previous rule query statement outputs data in the following format.

```
{
  "windspeed": 26.32020195042838
}
```

Important

This rule query statement ignores value updates other than the first in each batch. Each batch can contain up to 10 values. If you need to include the remaining values, you must set up a more complex solution to output asset property values to other services. For example, you can set up a rule with an AWS Lambda action to republish each value in the array to another topic, and set up another rule to query that topic and publish each value to the desired rule action.

Export data to Amazon S3 with asset property notifications

You can export incoming data from AWS IoT SiteWise to an Amazon S3 bucket in your account. You can back up your data in a format that you can use to create historical reports or to analyze your data with complex methods.

> **Note**
>
> AWS IoT SiteWise also supports cold tier storage that let you save data in a customer-managed Amazon S3 bucket. For more information about supported storage tiers, see Managing data storage.

AWS IoT SiteWise provides this feature as an AWS CloudFormation template. When you create a stack from the template, AWS CloudFormation creates the required AWS resources to stream incoming data from AWS IoT SiteWise to an S3 bucket.

Then, the S3 bucket receives all of your asset property data sent from AWS IoT SiteWise property value update messages. The S3 bucket also receives your asset metadata, which includes asset and property names and other information.

For more information about how to enable property value update messages for the asset properties to export to Amazon S3, see Interacting with other AWS services.

This feature stores your asset property data and asset metadata in the Apache Parquet format in Amazon S3. Parquet is a columnar data format that saves space and enables faster queries compared to row-oriented formats like JSON.

> **Note**
>
> When this feature retrieves asset metadata, it supports up to approximately 1,500 assets. This limitation applies only to asset metadata. This limitation doesn't apply to the number of assets supported when the feature exports asset property data.

Each resource's name includes a prefix that you can customize when you create the stack. Resources include the following:

- An Amazon S3 bucket
- AWS Lambda functions
- An AWS IoT Core rule
- AWS Identity and Access Management roles
- An Amazon Data Firehose stream

- An AWS Glue database

For a complete list, see Resources created from the template.

> **Important**
>
> You will be charged for the resources that this AWS CloudFormation template creates and consumes. These charges include data storage and data transfer for multiple AWS services.

Topics

Create the AWS CloudFormation stack

You must create a stack in AWS CloudFormation to export your asset data to Amazon S3.

To export data to Amazon S3

1. Open the AWS CloudFormation template and sign in to the AWS Management Console.
2. On the **Create stack** page, choose **Next** at the bottom of the page.
3. On the **Specify stack details** page, enter a **BucketName** for the S3 bucket that this template creates in order to receive asset data. This bucket name must be globally unique. For more information, see Rules for bucket naming in the *Amazon Simple Storage Service User Guide*.
4. (Optional) Change any of the template's other parameters:
   - **GlobalResourcePrefix** – A prefix for names of global resources, such as IAM roles, created from this template.
   - **LocalResourcePrefix** – A prefix for names of resources created from this template in the current Region.

> **Note**
>
> If you create this template multiple times, you should change the bucket name and resource prefix parameters in order to avoid resource name conflicts.

5. Choose **Next**.
6. On the **Configure stack options** page, choose **Next**.
7. At the bottom of the page, select the check box that says **I acknowledge that AWS CloudFormation might create IAM resources**.
8. Choose **Create stack**.

   The stack takes a few minutes to create. If the stack fails to create, your account might have insufficient permissions, or you might have entered a bucket name that already exists. Use the following steps to delete the stack and try again:

   a. Choose **Delete** in the upper-right corner.

      The stack takes a few minutes to delete.

      > **Note**
      >
      > AWS CloudFormation doesn't delete S3 buckets or CloudWatch log groups. You can delete these resources in the consoles for those services.

   b. If the stack fails to delete, choose **Delete** again.

   c. If the stack fails to delete again, follow the steps in the AWS CloudFormation console to skip the resources that failed to delete, and try again.
9. After the AWS CloudFormation stack creates successfully, follow the next procedure to explore your asset property data in Amazon S3.

> **Important**
>
> After you create the stack, you can see the new resources in your AWS account. The feature might stop working correctly if you delete or modify these resources. We recommend that

you don't modify these resources unless you want to stop sending data to the bucket or want to customize this feature.

View your data in Amazon S3

After you create the feature, you can view your asset property data and asset metadata in Amazon S3.

> **Note**
>
> Asset metadata updates every six hours. You might need to wait up to six hours to see asset metadata appear in the S3 bucket.

This feature stores asset property data in the following columns, where each row contains a data point:

- **type** – The type of property notification (`PropertyValueUpdate`).
- **asset_id** – The ID of the asset that received a data point.
- **asset_property_id** – The ID of the property that received a data point for the asset.
- **time_in_seconds** – The time at which the data was received, expressed in seconds in Unix epoch time.
- **offset_in_nanos** – The nanosecond offset from `timeInSeconds`.
- **asset_property_quality** – The quality of the data point: `GOOD`, `UNCERTAIN`, or `BAD`.
- **asset_property_value** – The value of the data point.
- **asset_property_data_type** – The data type of the asset property: `boolean`, `double`, `integer`, or `string`.

This feature stores asset metadata in the following columns, where each row contains an asset property:

- **asset_id** – The ID of the asset.
- **asset_name** – The name of the asset.
- **asset_model_id** – The ID of the asset's model.

- **asset_property_id** – The ID of the asset property.
- **asset_property_name** – The name of the asset property.
- **asset_property_data_type** – The data type of the asset property: `BOOLEAN`, `DOUBLE`, `INTEGER`, or `STRING`.
- **asset_property_unit** – The unit of the asset property.
- **asset_property_alias** – The alias of the asset property.

To view your AWS IoT SiteWise data in Amazon S3

1. Navigate to the Amazon S3 console.
2. From the list of buckets, choose the bucket with the name you chose when you created the template.
3. In the bucket, choose one of the following folders:
   - `asset-property-updates` – This folder contains asset property data exported from AWS IoT SiteWise.
   - `asset-metadata` – This folder contains asset details exported from AWS IoT SiteWise.
4. Choose the object that you want to view.
5. On the object's page, do the following:
   a. Choose the **Select from** tab.

      In this panel, you can preview records from Parquet files.
   b. For **File format**, choose **Parquet**.
   c. To show the contents of the file in JSON format, choose **Show file preview**.

> **Note**
>
> If new data doesn't appear in the bucket, check that you enabled property value update notifications for your asset properties. For more information, see Interacting with other AWS services.

For more information about how to analyze your asset data stored in the S3 bucket, see Analyze the exported data with Amazon Athena.

Analyze the exported data with Amazon Athena

After you have your asset property data in Amazon S3, you can use several AWS services to generate reports or analyze and query your data:

- Run SQL queries on your data using Amazon Athena.
- Perform big data analysis using Amazon EMR.
- Search and analyze your data using Amazon OpenSearch Service.

You can find other AWS services that can interact with your data in Amazon S3 listed under **Analytics** in the AWS Management Console.

> **Note**
>
> The stack creates an AWS Glue database to format asset property data. You can't query this database for asset data. Follow the steps in this section to create an AWS Glue database that you can query.

In this tutorial, you learn how to configure the prerequisites to use Amazon Athena and how to use Athena to run SQL queries on your exported AWS IoT SiteWise asset data. To query data with Athena, you must first populate the AWS Glue Data Catalog with your asset data. The Data Catalog contains databases and tables, and Athena can access data in the Data Catalog. You can create an AWS Glue crawler that regularly updates the Data Catalog with your exported asset data.

Topics

Configuring a crawler to populate the AWS Glue Data Catalog

AWS Glue crawlers crawl data stores to populate tables in the AWS Glue Data Catalog. In this procedure, you create and run an AWS Glue crawler for your S3 bucket that contains exported asset data. The crawler creates a table for asset property updates and a table for asset metadata. Then, you can perform SQL queries on these tables with Athena. For more information, see Populating the AWS Glue Data Catalog and Defining crawlers in the *AWS Glue Developer Guide*.

To create an AWS Glue crawler

1. Navigate to the AWS Glue console.
2. In the navigation pane, choose **Crawlers**.
3. Choose **Add crawler**.
4. On the **Add crawler** page, do the following:

   a. Enter a name for your crawler, such as **`IoTSiteWiseDataCrawler`**, and then choose **Next**.

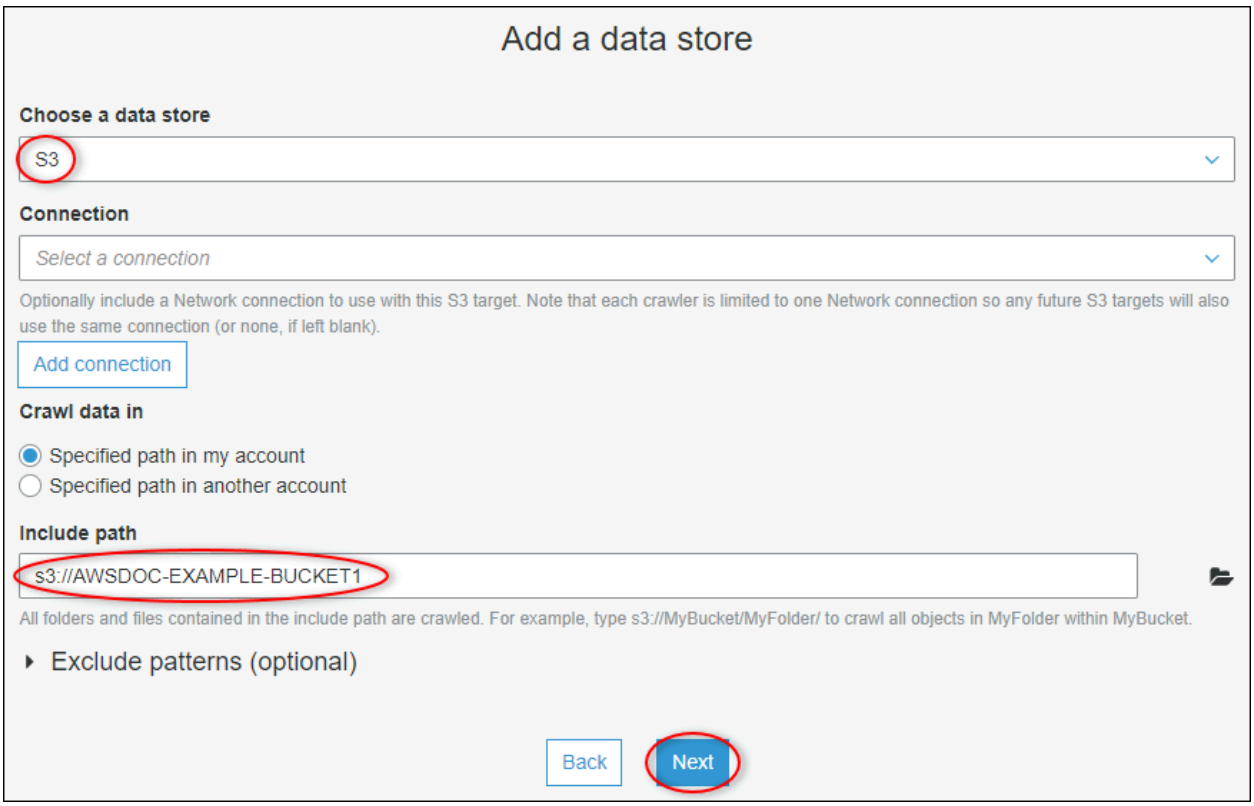
   b. For **Crawler source type**, choose **Data stores**, and then choose **Next**.

   c. On the **Add a data store page**, do the following:

      i. For **Choose a data store**, choose **S3**.

      ii. In **Include path**, enter **`s3://`*`DOC-EXAMPLE-BUCKET1`*** to add your asset data bucket as a data store. Replace *`DOC-EXAMPLE-BUCKET1`* with the bucket name that you chose when you created the stack.

      iii. Choose **Next**.



   d. On the **Add another data store** page, choose **No**, and then choose **Next**.

   e. On the **Choose an IAM role** page, do the following:

i. To create a new service role that allows AWS Glue to access the S3 bucket, choose **Create an IAM role**.

ii. Enter a suffix for your role's name, such as **`IoTSiteWiseDataCrawler`**.

iii. Choose **Next**.

f. For **Frequency**, choose **Hourly**, and then choose **Next**. The crawler updates the tables with new data each time it runs, so you can choose any frequency that fits your use case.

g. On the **Configure the crawler's output** page, do the following:

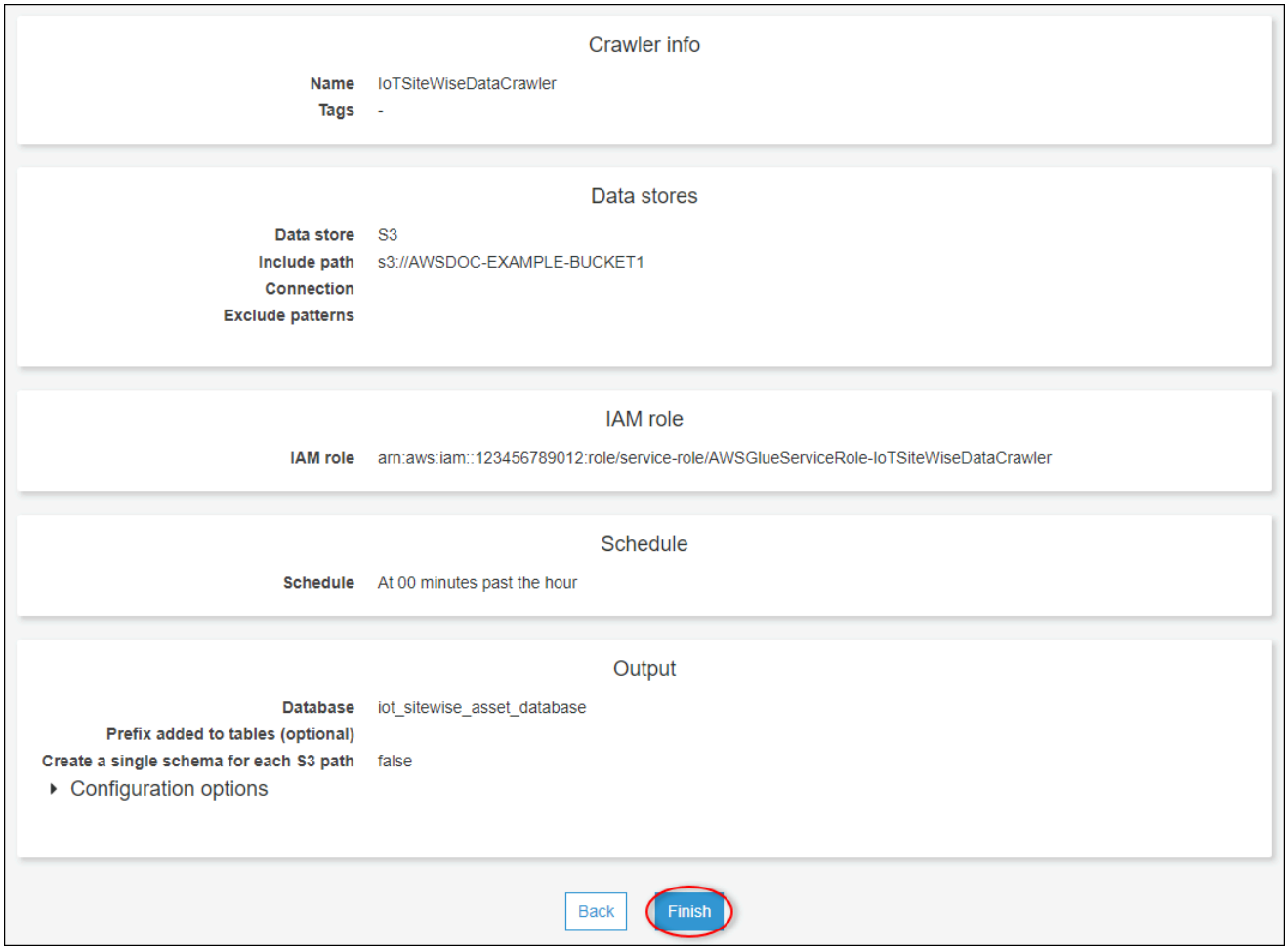
i. Choose **Add database** to create an AWS Glue database for your asset data.

ii. Enter a name for the database, such as **`iot_sitewise_asset_database`**.

iii. Choose **Create**.

iv. Choose **Next**.

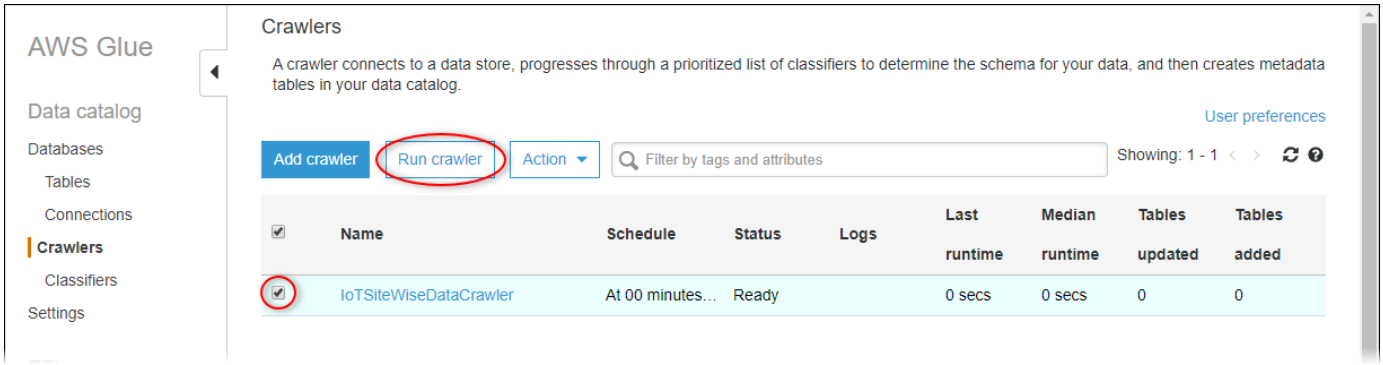
h. Review the crawler details, and then choose **Finish**.



By default, your new crawler doesn't immediately run. You must manually run it or wait until it runs on its configured schedule.

To run a crawler

1. On the **Crawlers** page, select the check box for your new crawler, and then choose **Run crawler**.



2. Wait until the crawler finishes and has a status of **Ready**.

   The crawler can take several minutes to run, and its status updates automatically.
3. In the navigation pane, choose **Tables**.

   You should see two new tables: **asset_metadata** and **asset_property_updates**.

Querying data with Athena

Athena automatically discovers your asset data tables in the AWS Glue Data Catalog. To perform queries on the intersection of these tables, you can create a view, which is a logical data table. For more information, see Working with views in the *Amazon Athena User Guide*.

After you create a view that combines asset property data and metadata, you can run queries that output property values with asset and property names attached. For more information, see Running SQL queries using Amazon Athena in the *Amazon Athena User Guide*.

To query asset data with Athena

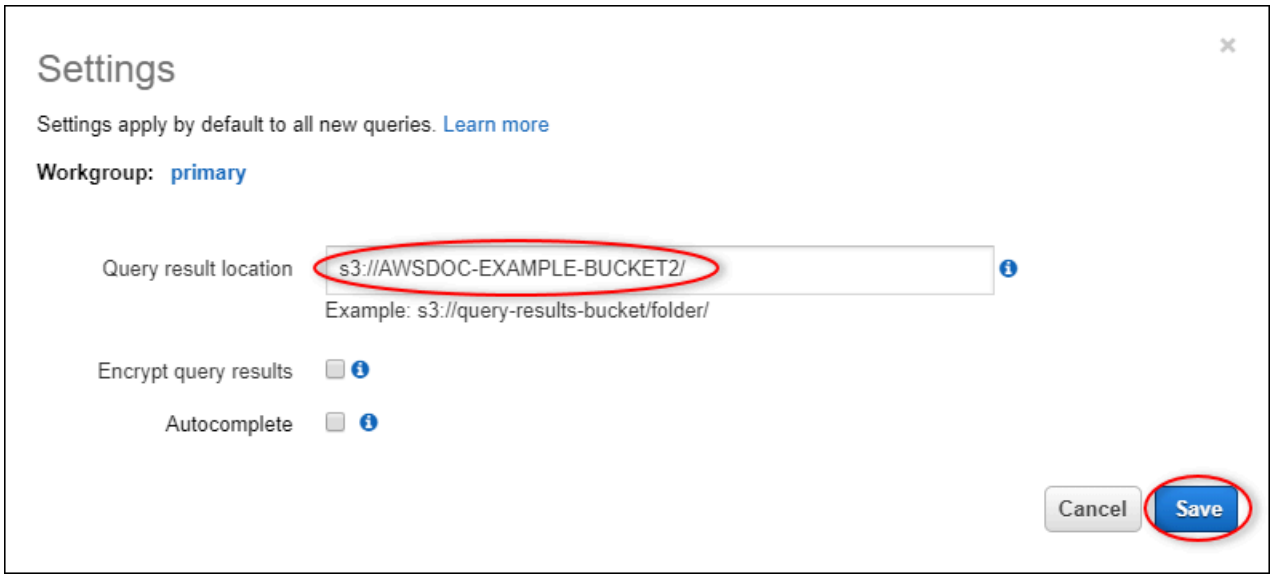
1. Navigate to the Athena console.

   If the **Getting started** page appears, choose **Get Started**.
2. If you're using Athena for the first time, complete the following steps to configure an S3 bucket for query results. Athena stores the results of your queries in this bucket.

> **Important**
>
> Use a different bucket than your asset data bucket, so the crawler that you created earlier doesn't crawl query results. We recommend that you create a bucket to use only for Athena query results. For more information, see How do I create an S3 bucket? in the *Amazon Simple Storage Service User Guide*.

a. Choose **Settings**.

b. In **Query result location**, enter the S3 bucket for Athena query results. The bucket must end with /.

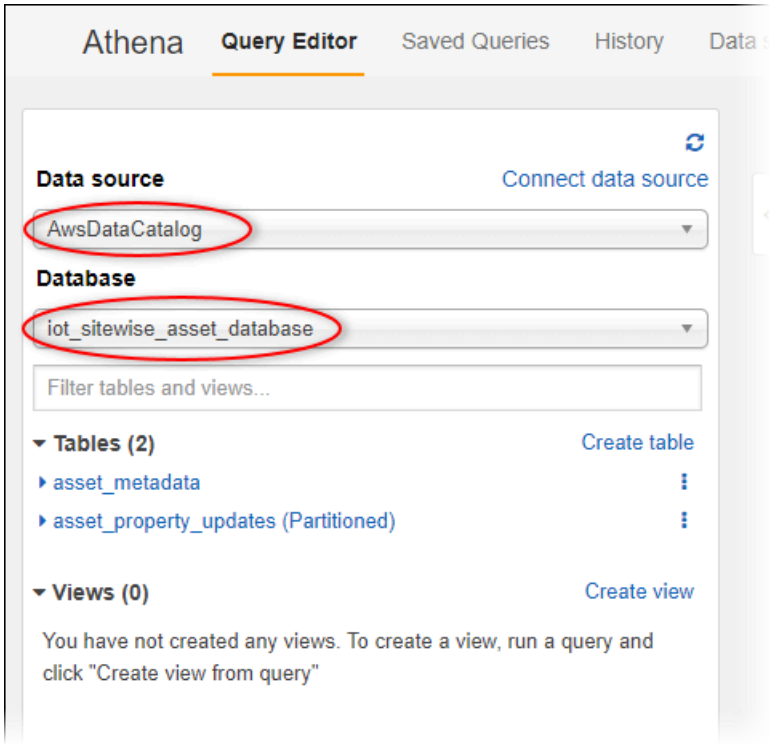


c. Choose **Save**.

3. The left panel contains the data source to query. Do the following:

a. For **Data source**, choose **AwsDataCatalog** to use the AWS Glue Data Catalog.

b. For **Database**, choose the AWS Glue database that you created with the crawler.



You should see two tables: **asset_metadata** and **asset_property_updates**.

4. To create a view from the combination of asset property data and metadata, enter the following query, and then choose **Run query**.

```
CREATE
        OR REPLACE VIEW iot_sitewise_asset_data AS
SELECT "from_unixtime"("time_in_seconds" + ("offset_in_nanos" / 1000000000))
 "timestamp",
         "metadata"."asset_name",
         "metadata"."asset_property_name",
         "data"."asset_property_value",
         "metadata"."asset_property_unit",
         "metadata"."asset_property_alias"
FROM ( "iot_sitewise_asset_database".asset_property_updates data
INNER JOIN "iot_sitewise_asset_database".asset_metadata metadata
    ON ( ("data"."asset_id" = "metadata"."asset_id")
        AND ("data"."asset_property_id" = "metadata"."asset_property_id") ) );
```

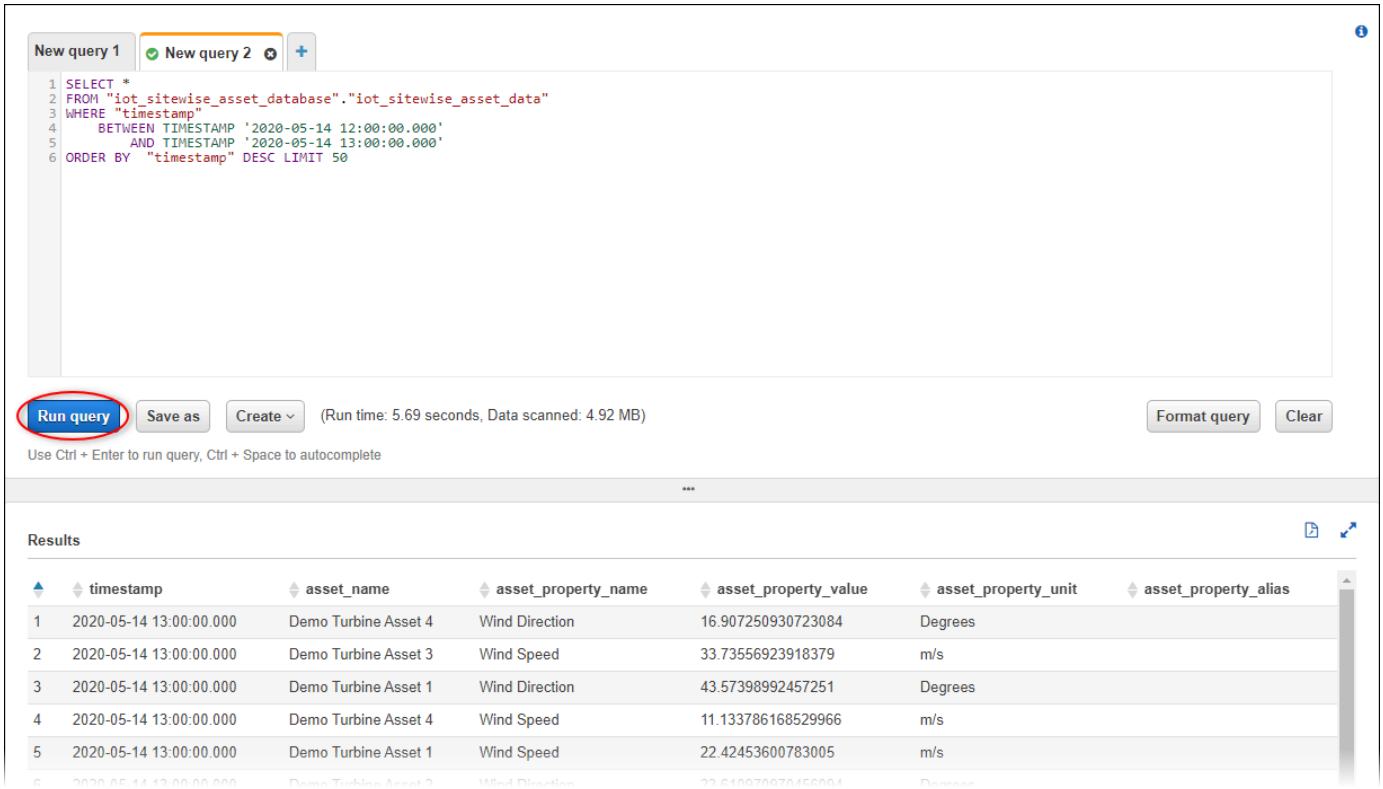
This query joins the asset property data and metadata tables on asset ID and property ID to create a view. You can run this query multiple times because it replaces the existing view if the view already exists.

5. To add a new query, choose the **+** icon.
6. To view a sample of asset data, enter the following query, and then choose **Run query**. Replace the timestamps with an interval for which your bucket has data.

```
SELECT *
FROM "iot_sitewise_asset_database"."iot_sitewise_asset_data"
WHERE "timestamp"
    BETWEEN TIMESTAMP '2020-05-14 12:00:00.000'
        AND TIMESTAMP '2020-05-14 13:00:00.000'
ORDER BY  "timestamp" DESC LIMIT 50;
```

This query outputs up to 50 data points between two timestamps, with the most recent entries shown first.

Your query output might look similar to the following results.



You can now run queries useful to your AWS IoT SiteWise application. For more information, see SQL reference for Amazon Athena in the *Amazon Athena User Guide*.

Resources created from the template

When you create a stack from the template, AWS CloudFormation creates the following resources. Most resources' names include a prefix that you can customize when you create the stack.

Resource name parameters

- `BucketName` – The name of the S3 bucket created from this template that receives asset data.
- `GlobalResourcePrefix` – A prefix for names of global resources created from this template. Defaults to `sitewise-export-to-s3`.
- `LocalResourcePrefix` – A prefix for names of resources created from this template in the current Region. Defaults to `sitewise_export_to_s3`.

Resources created by the AWS CloudFormation template

Resource	Description	Name
S3 bucket for processed data	This bucket contains two folders. One folder receives the flattened, formatted data from the Firehose delivery stream, and the other folder receives asset metadata.	`${BucketName}`
AWS Glue database	This database contains the AWS Glue table that this stack creates.	`${LocalResourcePrefix}_firehose_glue_database`
AWS Glue table	The Firehose delivery stream uses this table to format data to Parquet format.	`${LocalResourcePrefix}_firehose_glue_table`
AWS Lambda function that transforms data	This function flattens the array of values in property value notification messages sent from AWS IoT SiteWise.	`${LocalResourcePrefix}_lambda_transform_function`

Resource	Description	Name
IAM role for the transform Lambda function	This role allows Lambda to store runtime logs for the transform function.	`${GlobalResourcePrefix}-lambda-transform-role`
IAM policy for the transform Lambda function role	This policy allows Lambda to store execution logs for the transform function.	`${GlobalResourcePrefix}-lambda-transform-policy`
CloudWatch Logs log group for the transform function	This log group contains logs for the transform function.	`/aws/lambda/${LocalResourcePrefix}_lambda_transform_function`
Lambda function that collects asset metadata	This function retrieves details about assets in AWS IoT SiteWise and stores the details in an Amazon S3 bucket that this stack creates.	`${LocalResourcePrefix}_lambda_metadata_function`
Lambda layer for the metadata function	This layer provides an AWS SDK that contains AWS IoT SiteWise operations that the metadata function uses.	`${LocalResourcePrefix}_lambda_metadata_layer`
IAM role for the metadata Lambda function	This role allows Lambda to retrieve details about assets in AWS IoT SiteWise.	`${GlobalResourcePrefix}-lambda-metadata-role`
IAM policy for the metadata Lambda function role	This policy allows Lambda to retrieve details about assets in AWS IoT SiteWise.	`${GlobalResourcePrefix}-lambda-metadata-policy`
EventBridge scheduled event for the metadata Lambda function	This scheduled event runs the metadata Lambda every 6 hours to update the asset metadata bucket.	`${LocalResourcePrefix}-metadata-event`

Resource	Description	Name
CloudWatch Logs log group for the metadata function	This log group contains logs for the metadata function.	`/aws/lambda/${LocalResourcePrefix}_lambda_metadata_function`
AWS IoT rule	This rule queries property value notification messages and sends asset data to an Amazon Data Firehose delivery stream.	`${LocalResourcePrefix}_iot_topic_rule`
IAM role for the AWS IoT rule	This role allows AWS IoT to send data to the Firehose delivery stream.	`${GlobalResourcePrefix}-core-firehose-role`
IAM policy for the AWS IoT rule role	This policy allows AWS IoT to send data to the Firehose delivery stream.	`${GlobalResourcePrefix}-core-firehose-policy`
Firehose delivery stream	This delivery stream consumes data from the AWS IoT rule, flattens the data with a Lambda function, and delivers the data to Amazon S3.	`${LocalResourcePrefix}_firehose_delivery_stream`
IAM role for the delivery stream	This role allows Firehose to perform operations on the S3 bucket, AWS Glue table, Lambda functions, and CloudWatch Logs log group.	`${GlobalResourcePrefix}-firehose-delivery-role`
CloudWatch Logs log group for the delivery stream	This log group contains a log stream, `S3 Delivery`, that receives logs about the Firehose delivery stream.	`/aws/kinesisfirehose/${LocalResourcePrefix}_firehose_delivery_stream`

Integrating with Grafana

Grafana is a data visualization platform that you can use to visualize and monitor data in dashboards. In Grafana version 7.3.0 and later, you can use the AWS IoT SiteWise plugin to visualize your AWS IoT SiteWise asset data in Grafana dashboards. You can visualize data from multiple AWS sources (such as AWS IoT SiteWise, Amazon Timestream, and Amazon CloudWatch) and other data sources with a single Grafana dashboard.

You have two options to use the AWS IoT SiteWise plugin:

- **Local Grafana servers**

  You can set up the AWS IoT SiteWise plugin on a Grafana server that you manage. For more information about how to add and use the plugin, see the AWS IoT SiteWise Datasource README file on the GitHub website.

- **AWS Managed Service for Grafana**

  You can use the AWS IoT SiteWise plugin in the AWS Managed Service for Grafana (AMG). AMG manages Grafana servers for you so that you can visualize your data without having to build, package, or deploy any hardware or any other Grafana infrastructure. For more information, see the following topics in the *AWS Managed Service for Grafana User Guide*:

  - What is Amazon Managed Service for Grafana (AMG)?
  - Using the AWS IoT SiteWise data source

Example Example Grafana dashboard

The following Grafana dashboard visualizes the demo wind farm. You can access this demo dashboard on the Grafana Play website.

Integrating AWS IoT SiteWise and AWS IoT TwinMaker

Integrating with AWS IoT TwinMaker grants access to robust functionality in AWS IoT SiteWise, such as AWS IoT SiteWise data retrieval `ExecuteQuery` API and advanced asset search in the AWS IoT SiteWise console. To integrate the services and use these features, you must first enable the integration.

Topics

Enabling the integration

Administrators can use AWS JSON policies to specify who has access to what. That is, which *principal* can perform *actions* on what *resources*, and under what *conditions*. The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. For more information about AWS IoT SiteWise supported actions, see Actions defined by AWS IoT SiteWise in the *Service Authorization Reference*.

For more information about AWS IoT TwinMaker service-linked role, see Service-linked roles for AWS IoT TwinMaker in the *AWS IoT TwinMaker User Guide*.

Before you can integrate AWS IoT SiteWise and AWS IoT TwinMaker, you must grant the following permissions that allow AWS IoT SiteWise to integrate with an AWS IoT TwinMaker linked workspace:

- `iotsitewise:EnableSiteWiseIntegration` – Allows AWS IoT SiteWise to integrate with a linked AWS IoT TwinMaker workspace. This integration allows AWS IoT TwinMaker to read all your modeling information in AWS IoT SiteWise through an AWS IoT TwinMaker service-linked role. To enable this permission, add the following policy to your IAM role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:EnableSiteWiseIntegration"
      ],
      "Resource": "*"
    }
  ]
}
```

Integrating AWS IoT SiteWise and AWS IoT TwinMaker

To integrate AWS IoT SiteWise and AWS IoT TwinMaker, you must have the following:

- AWS IoT SiteWise service-linked role set up in your account
- AWS IoT TwinMaker service-linked role set up in your account

- AWS IoT TwinMaker workspace with ID `IoTSiteWiseDefaultWorkspace` in your account in the Region.

To integrate by using the AWS IoT SiteWise console

When you see the **Integration with AWS IoT TwinMaker** banner in the console, choose **Grant permission**. The prerequisites are created in your account.

To integrate by using the AWS CLI

To integrate AWS IoT SiteWise and AWS IoT TwinMaker by using the AWS CLI, enter the following commands:

1. Call `CreateServiceLinkedRole` with an `AWSServiceName` of `iotsitewise.amazonaws.com`.

```
aws iam create-service-linked-role --aws-service-name iotsitewise.amazonaws.com
```

2. Call `CreateServiceLinkedRole` with an `AWSServiceName` of `iottwinmaker.amazonaws.com`.

```
aws iam create-service-linked-role --aws-service-name iottwinmaker.amazonaws.com
```

3. Call `CreateWorkspace` with an `ID` of `IoTSiteWiseDefaultWorkspace`.

```
 aws iottwinmaker create-workspace --workspace-id IoTSiteWiseDefaultWorkspace
```

Detecting equipment anomalies with Amazon Lookout for Equipment

> **Note**
>
> Anomaly detection is only available in the Regions where Amazon Lookout for Equipment is available.

You can integrate AWS IoT SiteWise with Amazon Lookout for Equipment to gain insights about your industrial equipment through anomaly detection and predictive maintenance of industrial equipment. Lookout for Equipment is a machine learning (ML) service for monitoring industrial equipment that detects abnormal equipment behavior and identifies potential failures. With Lookout for Equipment, you can implement predictive maintenance programs and identify suboptimal equipment processes. For more information about Lookout for Equipment, see What is Amazon Lookout for Equipment? in the *Amazon Lookout for Equipment User Guide*.

When you create a prediction to train an ML model to detect anomalous equipment behavior, AWS IoT SiteWise sends asset property values to Lookout for Equipment to train an ML model to detect anomalous equipment behavior. To define a prediction definition on an asset model, you specify the IAM roles needed for Lookout for Equipment to access your data and the properties to send to Lookout for Equipment and send processed data to Amazon S3. For more information, see Creating asset models.

To integrate AWS IoT SiteWise and Lookout for Equipment, you'll perform the following high-level steps:

- Add a prediction definition on an asset model that outlines what properties you want to track. The prediction definition is a reusable collection of measurements, transforms, and metrics that is used to create predictions on the assets that are based on that asset model.
- Train the prediction based on historical data that you provide.
- Schedule inference, which tells AWS IoT SiteWise how often to run a specific prediction.

Once inference is scheduled, the Lookout for Equipment model monitors the data it receives from your equipment and looks for anomalies in equipment behavior. You can view and analyze the results in SiteWise Monitor, using the AWS IoT SiteWise GET API operations, or the Lookout for Equipment console. You can also create alarms using alarm detectors from the asset model to alert you about abnormal equipment behavior.

Topics

- Training a prediction (CLI)
- Starting or stopping inference on a prediction (CLI)

Adding a prediction definition (console)

To begin sending data collected by AWS IoT SiteWise to Lookout for Equipment, you must add an AWS IoT SiteWise prediction definition to an asset model.

To add a prediction definition to an AWS IoT SiteWise asset model

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Models** and select the asset model to which you want to add the prediction definition.
3. Choose **Predictions**.
4. Choose **Add prediction definition**.
5. Define details about the prediction definition.

   a. Enter a unique **Name** and a **Description** for your prediction definition. Choose the name thoughtfully because after you create the prediction definition, you can't change its name.

   b. Create or select an **IAM permissions role** that allows AWS IoT SiteWise to share your asset data with Amazon Lookout for Equipment. The role should have the following IAM and trust policies. For help creating the role, see Creating a role using custom trust policies (console).

      IAM policy

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Sid": "L4EPermissions",
            "Effect": "Allow",
            "Action": [
                "lookoutequipment:CreateDataset",
                "lookoutequipment:CreateModel",
                "lookoutequipment:CreateInferenceScheduler",
                "lookoutequipment:DescribeDataset",
                "lookoutequipment:DescribeModel",
                "lookoutequipment:DescribeInferenceScheduler",
                "lookoutequipment:ListInferenceExecutions",
```

```
                "lookoutequipment:StartDataIngestionJob",
                "lookoutequipment:StartInferenceScheduler",
                "lookoutequipment:UpdateInferenceScheduler",
                "lookoutequipment:StopInferenceScheduler"
            ],
            "Resource": [
                "arn:aws:lookoutequipment:Region:Account_ID:inference-
scheduler/IoTSiteWise_*",
                "arn:aws:lookoutequipment:Region:Account_ID:model/
IoTSiteWise_*",
                "arn:aws:lookoutequipment:Region:Account_ID:dataset/
IoTSiteWise_*"
            ]
        },
        {
            "Sid": "L4EPermissions2",
            "Effect": "Allow",
            "Action": [
                "lookoutequipment:DescribeDataIngestionJob"
            ],
            "Resource": "*"
        },
        {
            "Sid": "S3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:ListBucket",
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": ["arn:aws:s3:::iotsitewise-*"]
        },
        {
            "Sid": "IAMPermissions",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::Account_ID:role/Role_name"
        }
    ]
```

```
}
```

Trust policy

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Principal": {
                "Service": "iotsitewise.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "Account_ID"
                },
                "ArnEquals": {
                    "aws:SourceArn":
 "arn:aws:iotsitewise:Region:Account_ID:asset/*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lookoutequipment.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "Account_ID"
                },
                "ArnEquals": {
                    "aws:SourceArn":
 "arn:aws:lookoutequipment:Region:Account_ID:*"
                }
            }
        }
    ]
}
```

c. Choose **Next**.

6. Select data attributes (measurements, transforms, and metrics) that you want to send to Lookout for Equipment.

   a. (Optional) Select measurements.

   b. (Optional) Select transforms.

   c. (Optional) Select metrics.

   d. Choose **Next**.

7. Review your selections. To add the prediction definition to the asset model, on the summary page, choose **Add prediction definition**.

You can also **Edit** or **Delete** an existing prediction definition that has active predictions attached.

Training a prediction (console)

After you've added a prediction definition to an asset model, you can train the predictions that are on your assets.

To train a prediction in AWS IoT SiteWise

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**, and select the asset you want to monitor.
3. Choose **Predictions**.
4. Select the predictions that you want to train.
5. Under **Actions**, choose **Start training**, and do the following:

   a. Under **Prediction details**, select an IAM permissions role that allows AWS IoT SiteWise to share your asset data with Lookout for Equipment. If you need to create a new role, choose **Create a new role**.

   b. For **Training data settings**, enter a **Training data time range** to select which data to use to train the prediction.

   c. (Optional) For **Data labels**, provide an Amazon S3 bucket and prefix that holds your labeling data. For more information about labeling data, see Labeling your data in the *Amazon Lookout for Equipment User Guide*.

   d. Choose **Next**.

6. (Optional) If you want the prediction to be active as soon as it has completed training, under **Advanced settings**, select **Automatically activate the prediction after training**, and then do the following:

   a. Under **Input data**, for **Data upload frequency**, define how often data is uploaded, and for **Offset delay time**, define how much of a buffer to use.

   b. Choose **Next**.

7. Review the details of the prediction and choose **Save and start**.

Starting or stopping inference on a prediction (console)

> **Note**
>
> Lookout for Equipment charges apply to scheduled inferences with the data transferred between AWS IoT SiteWise and Lookout for Equipment. For more information, see Amazon Lookout for Equipment pricing.

If you added a prediction b"lookoutequipment:CreateDataset",ut did not choose to activate it after training, you must activate it for it to start monitoring your assets.

To start inference for a prediction

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Assets**, and select the asset the prediction is added to.
3. Choose **Predictions**.
4. Select the predictions that you want to activate.
5. Under **Actions**, choose **Start inference**, and do the following:

   a. Under **Input data**, for **Data upload frequency**, define how often data is uploaded, and for **Offset delay time**, define how much of a buffer to use.

   b. Choose **Save and start**.

To stop inference for a prediction

1. Navigate to the AWS IoT SiteWise console.

2. In the navigation pane, choose **Assets**, and select the asset the prediction is added to.
3. Choose **Predictions**.
4. Select the predictions that you want to stop.
5. Under **Actions**, choose **Stop inference**.

Adding a prediction definition (CLI)

To define a prediction definition on a new or existing asset model, you can use the AWS Command Line Interface (AWS CLI). After you define the prediction definition on the asset model, you train, and schedule inference for, a prediction on an asset in AWS IoT SiteWise to do anomaly detection with Lookout for Equipment.

Prerequisites

To complete these steps, you must have an asset model and at least one asset created. For more information, see Creating an asset model (AWS CLI) and Creating an asset (AWS CLI).

If you are new to AWS IoT SiteWise, you must call the `CreateBulkImportJob` API operation to import asset property values into AWS IoT SiteWise, which will be used to train the model. For more information, see Create a bulk import job (AWS CLI).

To add a prediction definition

1. Create a file called `asset-model-payload.json`. Follow the steps in these other sections to add your asset model's details to the file, but don't submit the request to create or update the asset model.

   - For more information about how to create an asset model, see Creating an asset model (AWS CLI)
   - For more information about how to update an existing asset model, see Updating an asset or component model (AWS CLI)
2. Add a Lookout for Equipment composite model (`assetModelCompositeModels`) to the asset model by adding the following code.

   - Replace *`Property`* with the ID of the properties that you want to include. To get those IDs, call `DescribeAssetModel`.
   - Replace *`RoleARN`* with the ARN of an IAM role that allows Lookout for Equipment to access your AWS IoT SiteWise data.

```
{
  ...
  "assetModelCompositeModels": [
    {
      "name": "L4Epredictiondefinition",
      "type": "AWS/L4E_ANOMALY",
      "properties": [
          {
            "name": "AWS/L4E_ANOMALY_RESULT",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_RESULT",
            "unit": "none",
            "type": {
              "measurement": {}
            }
          },
          {
            "name": "AWS/L4E_ANOMALY_INPUT",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_INPUT",
            "type": {
               "attribute": {
                 "defaultValue": "{\"properties\": [\"Property1\", \"Property2\"]}"
               }
            }
          },
          {
            "name": "AWS/L4E_ANOMALY_PERMISSIONS",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_PERMISSIONS",
            "type": {
              "attribute": {
                "defaultValue": "{\"roleArn\": \"RoleARN\"}"
              }
            }
          },
          {
            "name": "AWS/L4E_ANOMALY_DATASET",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_DATASET",
            "type": {
                "attribute": {}
```

```
            }
          },
          {
            "name": "AWS/L4E_ANOMALY_MODEL",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_MODEL",
            "type": {
              "attribute": {}
            }
          },
          {
            "name": "AWS/L4E_ANOMALY_INFERENCE",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_INFERENCE",
            "type": {
              "attribute": {}
            }
          },
          {
            "name": "AWS/L4E_ANOMALY_TRAINING_STATUS",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_TRAINING_STATUS",
            "type": {
              "attribute": {
                "defaultValue": "{}"
              }
            }
          },
          {
            "name": "AWS/L4E_ANOMALY_INFERENCE_STATUS",
            "dataType": "STRUCT",
            "dataTypeSpec": "AWS/L4E_ANOMALY_INFERENCE_STATUS",
            "type": {
              "attribute": {
                "defaultValue": "{}"
              }
            }
          }
        ]
}
```

3. Create the asset model or update the existing asset model. Do one of the following:

   - To create the asset model, run the following command:

```
aws iotsitewise create-asset-model --cli-input-json file://asset-model-
payload.json
```

- To update the existing asset model, run the following command. Replace *asset-model-id* with the ID of the asset model that you want to update.

```
aws iotsitewise update-asset-model \
  --asset-model-id asset-model-id \
  --cli-input-json file://asset-model-payload.json
```

After you run the command, note the `assetModelId` in the response.

Training a prediction and starting inference (CLI)

Now that the prediction definition is defined, you can train assets based on it and start inference. If you want to train your prediction but not start inference, skip to Training a prediction (CLI). To train the prediction and start inference on the asset, you'll need the `assetId` of the target resource.

To train and start inference of the prediction

1. Run the following command to find the `assetModelCompositeModelId` under `assetModelCompositeModelSummaries`. Replace *asset-model-id* with the ID of the asset model that you created in Updating an asset or component model (AWS CLI).

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. Run the following command to find the `actionDefinitionId` of the `TrainingWithInference` action. Replace *asset-model-id* with the ID used in previous step and replace *asset-model-composite-model-id* with the ID returned in the previous step.

```
aws iotsitewise describe-asset-model-composite-model \
  --asset-model-id asset-model-id \
  --asset-model-composite-model-id asset-model-composite-model-id \
```

3. Create a file called `train-start-inference-prediction.json` and add the following code, replacing the following:

- *asset-id* with the ID of the target asset
- *action-definition-id* with the ID of the TrainingWithInference action
- *StartTime* with the start of the training data, provided in epoch seconds
- *EndTime* with the end of the training data, provided in epoch seconds

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-Id",
  "actionPayload":{
    "stringValue": "{\"l4ETrainingWithInference\":{\"trainingWithInferenceMode
\":\"START\",\"trainingPayload\":{\"exportDataStartTime\":StartTime,
\"exportDataEndTime\":EndTime},\"inferencePayload\":{\"dataDelayOffsetInMinutes
\":0,\"dataUploadFrequency\":\"PT5M\"}}}"
  }
}
```

4. Run the following command to start training and inference:

```
aws iotsitewise execute-action --cli-input-json file://train-start-inference-
prediction.json
```

Training a prediction (CLI)

Now that the prediction definition is defined, you can train assets based on it. To train the prediction on the asset, you'll need the `assetId` of the target resource.

To train the prediction

1. Run the following command to find the `assetModelCompositeModelId` under `assetModelCompositeModelSummaries`. Replace *asset-model-id* with the ID of the asset model that you created in Updating an asset or component model (AWS CLI).

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. Run the following command to find the `actionDefinitionId` of the `Training` action. Replace *`asset-model-id`* with the ID used in previous step and replace *`asset-model-composite-model-id`* with the ID returned in the previous step.

```
aws iotsitewise describe-asset-model-composite-model \
  --asset-model-id asset-model-id \
  --asset-model-composite-model-id asset-model-composite-model-id \
```

3. Create a file called `train-prediction.json` and add the following code, replacing the following:

   - *`asset-id`* with the ID of the target asset
   - *`action-definition-id`* with the ID of the training action
   - *`StartTime`* with the start of the training data, provided in epoch seconds
   - *`EndTime`* with the end of the training data, provided in epoch seconds
   - (Optional) *`BucketName`* with the name of the Amazon S3 bucket that holds your label data
   - (Optional) *`Prefix`* with the prefix associated with the Amazon S3 bucket.

   > **Note**
   >
   > Include both the bucket name and prefix or neither.

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-Id",
  "actionPayload":{ "stringValue": "{\"l4ETraining\": {\"trainingMode\":
\"START\",\"exportDataStartTime\": StartTime, \"exportDataEndTime\": EndTime,
 \"labelInputConfiguration\": {\"bucketName\": \"BucketName\", \"prefix\":
 \"Prefix\"}}}"
}
}
```

4. Run the following command to start training:

```
aws iotsitewise execute-action --cli-input-json file://train-prediction.json
```

Before you can start inference, training must be completed. To check the status of the training, do one of the following:

- From the console, navigate to the asset the prediction is on.
- From the AWS CLI, call `BatchGetAssetPropertyValue` using the `propertyId` of the `trainingStatus` property.

Starting or stopping inference on a prediction (CLI)

Once the prediction is trained, you can start inference to tell Lookout for Equipment to start monitoring your assets. To start or stop inference, you'll need the `assetId` of the target resource.

To start inference

1. Run the following command to find the `assetModelCompositeModelId` under `assetModelCompositeModelSummaries`. Replace *`asset-model-id`* with the ID of the asset model that you created in Updating an asset or component model (AWS CLI).

   ```
   aws iotsitewise describe-asset-model \
     --asset-model-id asset-model-id \
   ```

2. Run the following command to find the `actionDefinitionId` of the `Inference` action. Replace *`asset-model-id`* with the ID used in previous step and replace *`asset-model-composite-model-id`* with the ID returned in the previous step.

   ```
   aws iotsitewise describe-asset-model-composite-model \
     --asset-model-id asset-model-id \
     --asset-model-composite-model-id asset-model-composite-model-id \
   ```

3. Create a file called `start-inference.json` and add the following code, replacing the following:

   - *`asset-id`* with the ID of the target asset
   - *`action-definition-id`* with the ID of the start inference action
   - *`Offset`* with the amount of buffer to use
   - *`Frequency`* with how often data is uploaded

   ```
   {
   ```

```
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-Id",
  "actionPayload":{ "stringValue": "{\"l4EInference\": {\"inferenceMode\":\"START
\",\"dataDelayOffsetInMinutes\": Offset, \"dataUploadFrequency\": \"Frequency\"}}"
}}
```

4. Run the following command to start inference:

```
aws iotsitewise execute-action --cli-input-json file://start-inference.json
```

To stop inference

1. Run the following command to find the `assetModelCompositeModelId` under `assetModelCompositeModelSummaries`. Replace *asset-model-id* with the ID of the asset model that you created in Updating an asset or component model (AWS CLI).

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. Run the following command to find the `actionDefinitionId` of the `Inference` action. Replace *asset-model-id* with the ID used in previous step and replace *asset-model-composite-model-id* with the ID returned in the previous step.

```
aws iotsitewise describe-asset-model-composite-model \
  --asset-model-id asset-model-id \
  --asset-model-composite-model-id asset-model-composite-model-id \
```

3. Create a file called `stop-inference.json` and add the following code, replacing the following:

   - *asset-id* with the ID of the target asset
   - *action-definition-id* with the ID of the start inference action

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
```

```
  "actionDefinitionId": "action-definition-Id",
  "actionPayload":{ "stringValue": "{\"l4EInference\":{\"inferenceMode\":\"STOP
\"}}"
}}
```

4. Run the following command to stop inference:

```
aws iotsitewise execute-action --cli-input-json file://stop-inference.json
```

Managing data storage

You can configure AWS IoT SiteWise to save your data in the following storage tiers:

Hot tier

The hot storage tier is an AWS IoT SiteWise managed time series storage. Hot tier is most effective for frequently accessed data, with low write-to-read latency. Data stored in the hot tier is used by industrial applications that need quick access to the latest values of measurements in your equipment. This includes applications that visualize real-time metrics with an interactive dashboard, or applications that monitor operations and launch alarms to identify performance issues.

By default, data ingested into AWS IoT SiteWise is stored in the hot tier. You can define a retention period for the hot tier, after which AWS IoT SiteWise moves data in the hot tier to either warm or cold tier storage, based on your configuration. For best performance and cost efficiency, set your hot tier retention period to be longer than the time taken to retrieve data often. This is used for real time metrics, alarms, and monitoring scenarios. If a retention period is not set, your data is stored indefinitely in the hot tier.

Warm tier

The warm storage tier is an AWS IoT SiteWise managed tier that's effective for cost-efficient storage of historical data. It's best used to retrieve large volumes of data with medium write-to-read latency characteristics. Use the warm tier to store historical data that's needed for large workloads. For example, it's used for data retrieval for analytics, business intelligence applications (BI), reporting tools, and training of machine learning (ML) models. If you enable the cold storage tier, you can define a warm tier retention period. After the retention period ends, AWS IoT SiteWise deletes data from the warm tier.

Cold tier

The cold storage tier uses an Amazon S3 bucket to store data that's rarely used. With cold tier enabled, AWS IoT SiteWise replicates the time series, including measurements, metrics, transforms and aggregates, and asset model definitions every 6 hours. Cold tier is used to store data that tolerates high read latency for historical reports and backups.

Topics

Configure storage settings

You can configure storage settings to opt in to service managed warm tier storage, and also to replicate data to the cold tier. To learn more about the retention period for the warm and hot tier, see Data retention impact. While configuring the storage settings, do the following:

- **Hot tier retention** — Set a retention period for how long your data is stored in the hot tier before it's deleted, and moved to the service managed warm tier storage or cold tier storage based on your storage settings. AWS IoT SiteWise will delete any data in the hot tier that existed before the retention period ends. If you don't set a retention period, your data is stored indefinitely in the hot tier.
- **Warm tier retention** — Set a retention period for how long your data is stored in the warm tier before it's deleted from AWS IoT SiteWise storage and moved to the customer managed cold tier storage. AWS IoT SiteWise deletes any data from the warm tier that existed before the retention period ends. If a retention period is not set, your data is stored indefinitely in the warm tier.

> **Note**
>
> To improve query performance, set a hot tier retention period with warm tier storage.

Impact of data retention in hot and warm tier storage

- When you decrease the retention period of the hot tier storage, data is permanently moved from the hot tier to the warm or cold tier. When you decrease the retention period of the warm tier, data is moved to the cold tier, and permanently deleted from the warm tier.
- When you increase the retention period of the hot or warm tier storage, the change affects data that's sent to AWS IoT SiteWise from then on. AWS IoT SiteWise does not retrieve data from the warm or cold storage to populate the hot tier. For example, if the retention period of the hot tier storage is initially set for 30 days and then increased to 60 days, it takes 30 days for the hot tier storage to contain 60 days worth of data.

Topics

Configure storage settings for warm tier (console)

The following procedure shows you how to configure the storage settings to replicate data to the warm tier in the AWS IoT SiteWise console.

To configure storage settings in the console

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, under **Settings**, choose **Storage**.
3. In the upper-right corner, choose **Edit**.
4. On the **Edit storage** page, do the following:
5. For **Hot tier settings**, do the following:

   - If you want to set a retention period for how long your data is stored in the hot tier before it's deleted, and moved to the service managed warm tier storage, choose **Enable retention period**.
   - To configure a retention period, enter a whole number and choose a unit. The retention period must be greater than or equal to 30 days.

   AWS IoT SiteWise deletes any data in the hot tier that's older than the retention period. If you don't set a retention period, your data is stored indefinitely.
6. (Recommended) For **Warm tier settings**, do the following:

   - To opt in to warm tier storage, select **I confirm to the opt-in of warm tier storage** to opt in for the warm tier storage.
   - (Optional) To configure a retention period, enter a whole number and choose a unit. The retention period must be greater than or equal to 365 days.

AWS IoT SiteWise deletes data in the warm tier that existed earlier than the retention period. If you don't set a retention period, your data is stored indefinitely.

> **Note**
>
> - When you opt in for warm tier, the configuration displays once only.
> - To set hot tier retention, you must have either warm or cold tier storage. For cost efficiency and historical data retrieval, AWS IoT SiteWise recommends that you store long term data in the warm tier.
> - To set warm tier retention, you must have cold tier storage.

7. Choose **Save** to save your storage settings.

In the **AWS IoT SiteWise storage** section, the **Warm tier storage** is in one of these states:

- **Enabled** – If your data existed before the hot tier retention period, AWS IoT SiteWise moves the data to the warm tier."
- **Disabled** – The warm tier storage is disabled.

Configure storage settings for warm tier (AWS CLI)

You can configure storage settings to move data to the warm tier by using the AWS CLI and the following commands.

To prevent overriding the existing configuration, retrieve the current storage configuration information by running the following command:

```
aws iotsitewise describe-storage-configuration
```

Example response without existing cold tier configuration

```
{
          "storageType": "SITEWISE_DEFAULT_STORAGE",
          "disassociatedDataStorage": "ENABLED",
          "configurationStatus": {
```

```
                "state": "ACTIVE"
            },
            "lastUpdateDate": "2021-10-14T15:53:35-07:00",
            "warmTier": "DISABLED"
}
```

Example response with existing cold tier configuration

```
{
      "storageType": "MULTI_LAYER_STORAGE",
          "multiLayerStorage": {
            "customerManagedS3Storage": {
            "s3ResourceArn": "arn:aws:s3:::bucket-name/prefix/",
            "roleArn": "arn:aws:iam::aws-account-id:role/role-name"
            }
          },
      "disassociatedDataStorage": "ENABLED",
      "retentionPeriod": {
      "numberOfDays": retention-in-days
      },
       "configurationStatus": {
       "state": "ACTIVE"
      },
      "lastUpdateDate": "2023-10-25T15:59:46-07:00",
      "warmTier": "DISABLED"
}
```

Configure storage settings for warm tier with AWS CLI

Run the following command to configure the storage settings. Replace `file-name` with the name of the file that contains the AWS IoT SiteWise storage configuration.

```
aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
```

Example AWS IoT SiteWise configuration with hot and warm tier

```
{
            "storageType": "SITEWISE_DEFAULT_STORAGE",
            "disassociatedDataStorage": "ENABLED",
            "warmTier": "ENABLED",
            "retentionPeriod": {
```

```
                "numberOfDays": hot-tier-retention-in-days
              }

}
```

`hot-tier-retention-in-days` must be a whole number greater than or equal to 30 days.

Example response

```
{
            "storageType": "SITEWISE_DEFAULT_STORAGE",
            "configurationStatus": {
            "state": "UPDATE_IN_PROGRESS"
            }
}
```

If you have cold tier storage enabled, see Configure storage settings with AWS CLI and existing cold tier.

Configure storage settings with AWS CLI and existing cold tier

Configure storage settings using AWS CLI with existing cold tier storage

- Run the following command to configure the storage settings. Replace *file-name* with the name of the file that contains the AWS IoT SiteWise storage configuration.

  ```
  aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
  ```

  Example AWS IoT SiteWise storage configuration

  - Replace *bucket-name* with your Amazon S3 bucket name.
  - Replace *prefix* with your Amazon S3 prefix.
  - Replace *aws-account-id* with your AWS account ID.
  - Replace *role-name* with the name of the Amazon S3 access role that allows AWS IoT SiteWise to send data to Amazon S3.
  - Replace *hot-tier-retention-in-days* with a whole number greater than or equal to 30 days.
  - Replace *warm-tier-retention-in-days* with a whole number greater than or equal to 365 days.

> **Note**
>
> AWS IoT SiteWise will delete any data in the warm tier that's older than the retention period of the cold tier. If you don't set a retention period, your data is stored indefinitely.

```
{
      "storageType": "MULTI_LAYER_STORAGE",
        "multiLayerStorage": {
          "customerManagedS3Storage": {
              "s3ResourceArn": "arn:aws:s3:::bucket-name/prefix/",
              "roleArn": "arn:aws:iam::aws-account-id:role/role-name"
              }
          },
    "disassociatedDataStorage": "ENABLED",
    "retentionPeriod": {
      "numberOfDays": hot-tier-retention-in-days
    },
    "warmTier": "ENABLED",
    "warmTierRetentionPeriod": {
      "numberOfDays": warm-tier-retention-in-days
    }
}
```

Example response

```
{
      "storageType": "MULTI_LAYER_STORAGE",
      "configurationStatus": {
        "state": "UPDATE_IN_PROGRESS"
       }
}
```

Configure storage settings for cold tier (console)

The following procedure shows you how to configure the storage settings to replicate data to the cold tier in the AWS IoT SiteWise console.

To configure storage settings in the console

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, under **Settings**, choose **Storage**.
3. In the upper-right corner, choose **Edit**.
4. On the **Edit storage** page, do the following:

   a. For **Storage settings**, choose **Enable cold tier storage**. The cold tier storage is disabled by default.

   b. For **S3 bucket location**, enter the name of an existing Amazon S3 bucket and a prefix.

   > **Note**
   >
   > - Amazon S3 uses the prefix as a folder name in the Amazon S3 bucket. The prefix must have 1-255 characters and end with a forward slash (/). Your AWS IoT SiteWise data is saved in this folder.
   > - If you don't have an Amazon S3 bucket, choose **View**, and then create one in the Amazon S3 console. For more information, see Create your first S3 bucket in the *Amazon S3 User Guide*.

   c. For **S3 access role**, do one of the following:

   - Choose **Create a role from an AWS managed template**, AWS automatically creates an IAM role that allows AWS IoT SiteWise to send data to Amazon S3.
   - Choose **Use an existing role**, and then choose the role that you created from the list.

   > **Note**
   >
   > - You must use the same Amazon S3 bucket name for the **S3 bucket location** that you used in the previous step and in your IAM policy.
   > - Make sure that your role has the permissions shown in the following example.
   >
   > **Example permissions policy:**
   >
   > ```
   > {
   >     "Version": "2012-10-17",
   >     "Statement": [
   > ```

```
            {
                "Effect": "Allow",
                "Action": [
                    "s3:PutObject",
                    "s3:GetObject",
                    "s3:DeleteObject",
                    "s3:GetBucketLocation",
                    "s3:ListBucket"
                ],
                "Resource": [
                    "arn:aws:s3:::bucket-name",
                    "arn:aws:s3:::bucket-name/*"
                ]
            }
        ]
    }
```

Replace *bucket-name* with the name of your Amazon S3 bucket.

d. To setup hot tier, see Step 5 in Configure storage settings for warm tier (console).

e. (Optional) For **AWS IoT Analytics integration**, do the following.

   i. If you want to use AWS IoT Analytics to query your data, choose **Enabled AWS IoT Analytics data store**.

   ii. AWS IoT SiteWise generates a name for your data store or you can enter a different name.

   AWS IoT SiteWise automatically creates a data store in AWS IoT Analytics to save your data. To query the data, you can use AWS IoT Analytics to create datasets. For more information, see Working with AWS IoT SiteWise data in the *AWS IoT Analytics User Guide*.

f. Choose **Save**.

In the **AWS IoT SiteWise storage** section, the **Cold tier storage** can be one of the following values:

- **Enabled** – AWS IoT SiteWise replicates your data to the specified Amazon S3 bucket.
- **Enabling** – AWS IoT SiteWise is processing your request to enable the cold tier storage. This process can take several minutes to complete.

- **Enable_Failed** – AWS IoT SiteWise couldn't process your request to enable the cold tier storage. If you enabled AWS IoT SiteWise to send logs to Amazon CloudWatch Logs, you can use these logs to troubleshoot issues. For more information, see Monitoring with Amazon CloudWatch Logs.
- **Disabled** – The cold tier storage is disabled.

Configure storage settings for cold tier (AWS CLI)

The following procedure shows you how to configure the storage settings to replicate data to the cold tier using AWS CLI.

To configure storage settings using AWS CLI

1. To export data to an Amazon S3 bucket in your account, run the following command to configure the storage settings. Replace *file-name* with the name of the file that contains the AWS IoT SiteWise storage configuration.

   ```
   aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
   ```

   Example AWS IoT SiteWise storage configuration

   - Replace *bucket-name* with your Amazon S3 bucket name.
   - Replace *prefix* with your Amazon S3 prefix.
   - Replace *aws-account-id* with your AWS account ID.
   - Replace *role-name* with the name of the Amazon S3 access role that allows AWS IoT SiteWise to send data to Amazon S3.
   - Replace *retention-in-days* with a whole number than is greater than or equal to 30 days.

   ```
   {
         "storageType": "MULTI_LAYER_STORAGE",
         "multiLayerStorage": {
             "customerManagedS3Storage": {
                 "s3ResourceArn": "arn:aws:s3:::bucket-name/prefix/",
                 "roleArn": "arn:aws:iam::aws-account-id:role/role-name"
             }
         },
   ```

```
    "retentionPeriod": {
        "numberOfDays": retention-in-days,
        "unlimited": false
    }
}
```

Note

- You must use the same Amazon S3 bucket name in the AWS IoT SiteWise storage configuration and IAM policy.
- Make sure that your role has the permissions shown in the following example.

  Example permissions policy:

  ```
  {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "s3:PutObject",
                    "s3:GetObject",
                    "s3:DeleteObject",
                    "s3:GetBucketLocation",
                    "s3:ListBucket"
                ],
                "Resource": [
                    "arn:aws:s3:::bucket-name",
                    "arn:aws:s3:::bucket-name/*"
                ]
            }
        ]
    }
  ```

  Replace *bucket-name* with the name of your Amazon S3 bucket.

Example response

```
{
```

```
    "storageType": "MULTI_LAYER_STORAGE",
    "retentionPeriod": {
        "numberOfDays": 100,
        "unlimited": false
    },
    "configurationStatus": {
        "state": "UPDATE_IN_PROGRESS"
    }
}
```

> **Note**
>
> It can take a few minutes for AWS IoT SiteWise to update the storage configuration.

2. To retrieve the storage configuration information, run the following command.

```
aws iotsitewise describe-storage-configuration
```

Example response

```
{
      "storageType": "MULTI_LAYER_STORAGE",
      "multiLayerStorage": {
          "customerManagedS3Storage": {
              "s3ResourceArn": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/torque/",
              "roleArn": "arn:aws:iam::123456789012:role/SWAccessS3Role"
          }
      },
      "retentionPeriod": {
          "numberOfDays": 100,
          "unlimited": false
      },
      "configurationStatus": {
          "state": "ACTIVE"
      },
      "lastUpdateDate": "2021-03-30T15:54:14-07:00"
  }
```

3. To stop exporting data to the Amazon S3 bucket, run the following command to configure storage settings.

```
aws iotsitewise put-storage-configuration --storage-type SITEWISE_DEFAULT_STORAGE
```

> **Note**
>
> By default, your data is only stored in the hot tier of AWS IoT SiteWise.

Example response

```
{
      "storageType": "SITEWISE_DEFAULT_STORAGE",
      "configurationStatus": {
          "state": "UPDATE_IN_PROGRESS"
      }
  }
```

4. To retrieve the storage configuration information, run the following command.

```
aws iotsitewise describe-storage-configuration
```

Example response

```
{
      "storageType": "SITEWISE_DEFAULT_STORAGE",
      "configurationStatus": {
          "state": "ACTIVE"
      },
      "lastUpdateDate": "2021-03-30T15:57:14-07:00"
  }
```

(Optional) Create an AWS IoT Analytics data store (AWS CLI)

An AWS IoT Analytics data store is a scalable and queryable repository that receives and stores data. You can use the AWS IoT SiteWise console or AWS IoT Analytics APIs to create an AWS IoT Analytics data store to save your AWS IoT SiteWise data. To query the data, you create datasets by using AWS IoT Analytics. For more information, see Working with AWS IoT SiteWise data in the *AWS IoT Analytics User Guide*.

The following steps use AWS CLI to create a data store in AWS IoT Analytics.

To create a data store, run the following command. Replace *file-name* with the name of the file that contains the data store configuration.

```
aws iotanalytics create-datastore --cli-input-json file://file-name.json
```

Note

- You must specify the name of an existing Amazon S3 bucket. If you don't have an Amazon S3 bucket, create one first. For more information, see Create your first S3 bucket in *Amazon S3 User Guide*.
- You must use the same Amazon S3 bucket name in the AWS IoT SiteWise storage configuration, IAM policy, and AWS IoT Analytics data store configuration.

Example AWS IoT Analytics data store configuration

Replace *data-store-name* and *s3-bucket-name* with your AWS IoT Analytics data store name and Amazon S3 bucket name.

```
{
      "datastoreName": "data-store-name",
      "datastoreStorage": {
          "iotSiteWiseMultiLayerStorage": {
              "customerManagedS3Storage": {
                  "bucket": "s3-bucket-name"
              }
          }
      },
      "retentionPeriod": {
          "numberOfDays": 90
      }
  }
```

Example response

```
{
      "datastoreName": "datastore_IoTSiteWise_demo",
```

```
      "datastoreArn": "arn:aws:iotanalytics:us-west-2:123456789012:datastore/
datastore_IoTSiteWise_demo",
      "retentionPeriod": {
          "numberOfDays": 90,
          "unlimited": false
      }
  }
```

Troubleshoot storage settings

Use the following information to troubleshoot and resolve issues with the storage configuration.

Issues

- Error: Bucket doesn't exist
- Error: Access denied to Amazon S3 path
- Error: Role ARN can't be assumed
- Error: Failed to access cross-Region Amazon S3 bucket

Error: Bucket doesn't exist

Solution: AWS IoT SiteWise couldn't find your Amazon S3 bucket. Make sure you enter the name of an existing Amazon S3 bucket in the current Region.

Error: Access denied to Amazon S3 path

Solution: AWS IoT SiteWise couldn't access your Amazon S3 bucket. Do the following:

- Make sure that you use the same Amazon S3 bucket that you specified in the IAM policy.
- Make sure that your role has the permissions shown in the following example.

Example permissions policy

```
{
      "Version": "2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": [
```

```
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject",
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

Replace *bucket-name* with the name of your Amazon S3 bucket.

Error: Role ARN can't be assumed

Solution: AWS IoT SiteWise couldn't assume the IAM role on your behalf. Make sure that your role trusts the following service: `iotsitewise.amazonaws.com`. For more information, see I can't assume a role see *IAM User Guide*.

Error: Failed to access cross-Region Amazon S3 bucket

Solution: The Amazon S3 bucket that you specified is in a different AWS Region. Make sure that your Amazon S3 bucket and AWS IoT SiteWise assets are in the same Region.

File paths and schemas of data saved in the cold tier

AWS IoT SiteWise stores your data in the cold tier by replicating time series, including measurements, metrics, transforms and aggregates, and also asset and asset model definitions. The following describes the file paths and schemas of data that is sent to the cold tier.

Topics

- Storage data index files

Equipment data (measurements)

AWS IoT SiteWise exports equipment data (measurements) to the cold tier once every six hours. Raw data is saved in the cold tier in the Apache AVRO (`.avro`) format.

File path

AWS IoT SiteWise stores equipment data (measurements) in the cold tier using the following template.

```
{keyPrefix}/raw/startYear={startYear}/startMonth={startMonth}/startDay={startDay}/
seriesBucket={seriesBucket}/raw_{timeseriesId}_{startTimestamp}_{quality}.avro
```

Every file path to raw data in Amazon S3 contains the following components.

Path component	Description
`keyPrefix`	The Amazon S3 prefix that you specified in the AWS IoT SiteWise storage configuration. Amazon S3 uses the prefix as a folder name in the bucket.
`raw`	The folder that stores time series data from equipment (measurements). The `raw` folder is saved in the prefix folder.
`seriesBucket`	A hexadecimal number between 00 and ff. This number is derived from `timeSeriesId` . This partition is used to increase throughput when AWS IoT SiteWise writes to the cold tier. When you use Amazon Athena to run queries, you can use the partition for fine-grain partitioning to improve query performance. `seriesBucket` and `timeSeriesBucket` in the asset metadata are the same number.

Path component	Description
`startYear`	The year of the exclusive start time associated with the time series data.
`startMonth`	The month of the exclusive start time associated with the time series data.
`startDay`	The day of the month of the exclusive start time associated with the time series data.
`fileName`	The file name uses the underscore (_) character as a delimiter to separate the following: • The `raw` prefix. • The `timeSeriesId` value. • The epoch timestamp of the exclusive start time associated with the time series data. • The quality of the data. Valid values: `GOOD`, `BAD`, and `UNCERTAIN`. For more information, see AssetPropertyValue in the *AWS IoT SiteWise API Reference*. The file is saved in the `.avro` format by using the Snappy compression.

Example file path to raw data in the cold tier

```
keyPrefix/raw/startYear=2021/startMonth=1/startDay=2/seriesBucket=a2/
raw_7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a_1609577700_GOOD.avro
```

Fields

The schema of raw data that is exported to the cold tier contains the following fields.

Field name	Supported types	Default type	Description
`seriesId`	`string`	N/A	The ID that identifies the time series data from equipment (measurements). You can use this field to join raw data and asset metadata in queries.
`timeInSeconds`	`long`	N/A	The timestamp date, in seconds, in the Unix epoch format. Fractional nanosecond data is provided by `offsetInNanos`.
`offsetInNanos`	`long`	N/A	The nanosecond offset from `timeInSeconds`.
`quality`	`string`	N/A	The quality of the time series value.
`doubleValue`	`double` or `null`	`null`	Time series data of type double (floating point number).
`stringValue`	`string` or `null`	`null`	Time series data of type string (sequence of characters).
`integerValue`	`int` or `null`	`null`	Time series data of type integer (whole number).

Field name	Supported types	Default type	Description
booleanValue	boolean or null	null	Time series data of type Boolean (true or false).
jsonValue	string or null	null	Time series data of type JSON (complex data types stored as a string).
recordVersion	long or null	null	The version number for the record. You can use the version number to select the latest record. Newer records have larger version numbers.

Example raw data in the cold tier


```
{"seriesId":"e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeInSeconds":1625675887,"offsetInNanos":0,"quality":"GOOD","doubleValue":
{"double":0.75},"stringValue":null,"integerValue":null,"booleanValue":null,"jsonValue":null,"re
  {"seriesId":"e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeInSeconds":1625675889,"offsetInNanos":0,"quality":"GOOD","doubleValue":
{"double":0.69},"stringValue":null,"integerValue":null,"booleanValue":null,"jsonValue":null,"re
  {"seriesId":"e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeInSeconds":1625675890,"offsetInNanos":0,"quality":"GOOD","doubleValue":
{"double":0.66},"stringValue":null,"integerValue":null,"booleanValue":null,"jsonValue":null,"re
  {"seriesId":"e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeInSeconds":1625675891,"offsetInNanos":0,"quality":"GOOD","doubleValue":
{"double":0.92},"stringValue":null,"integerValue":null,"booleanValue":null,"jsonValue":null,"re
  {"seriesId":"e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeInSeconds":1625675892,"offsetInNanos":0,"quality":"GOOD","doubleValue":
{"double":0.73},"stringValue":null,"integerValue":null,"booleanValue":null,"jsonValue":null,"re
```

Metrics, transforms, and aggregates

AWS IoT SiteWise exports metrics, transforms, and aggregates to the cold tier once every six hours. Metrics, transforms, and aggregates are saved in the cold tier in the Apache AVRO (`.avro`) format.

File path

AWS IoT SiteWise stores metrics, transforms, and aggregates in the cold tier using the following template.

```
{keyPrefix}/agg/startYear={startYear}/startMonth={startMonth}/startDay={startDay}/
seriesBucket={seriesBucket}/agg_{timeseriesId}_{startTimestamp}_{quality}.avro
```

Every file path to metrics, transforms, and aggregates in Amazon S3 contains the following components.

Path component	Description
`keyPrefix`	The Amazon S3 prefix that you specified in the AWS IoT SiteWise storage configuration. Amazon S3 uses the prefix as a folder name in the bucket.
`agg`	The folder that stores time series data from metrics. The `agg` folder is saved in the prefix folder.
`seriesBucket`	A hexadecimal number between 00 and ff. This number is derived from `timeSeriesId` . This partition is used to increase throughput when AWS IoT SiteWise writes to the cold tier. When you use Amazon Athena to run queries, you can use the partition for fine-grain partitioning to improve query performance. `seriesBucket` and `timeSeriesBucket` in the asset metadata are the same number.

Path component	Description
`startYear`	The year of the exclusive start time associated with the time series data.
`startMonth`	The month of the exclusive start time associated with the time series data.
`startDay`	The day of the month of the exclusive start time associated with the time series data.
`fileName`	The file name uses the underscore (_) character as a delimiter to separate the following: • The `raw` prefix. • The `timeSeriesId` value. • The epoch timestamp of the exclusive start time associated with the time series data. • The quality of the data. Valid values: `GOOD`, `BAD`, and `UNCERTAIN`. For more information, see AssetPropertyValue in the *AWS IoT SiteWise API Reference*. The file is saved in the `.avro` format by using the Snappy compression.

Example file path to metrics in the cold tier

```
keyPrefix/agg/startYear=2021/startMonth=1/startDay=2/seriesBucket=a2/
agg_7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a_1609577700_GOOD.avro
```

Fields

The schema of metrics, transforms, and aggregates that are exported to the cold tier contains the following fields.

Field name	Supported types	Default type	Description
`seriesId`	`string`	N/A	The ID that identifies the time series data from equipment, metrics, or transforms. You can use this field to join raw data and asset metadata in queries.
`timeInSeconds`	`long`	N/A	The timestamp date, in seconds, in the Unix epoch format. Fractional nanosecond data is provided by `offsetInNanos`.
`offsetInNanos`	`long`	N/A	The nanosecond offset from `timeInSeconds`.
`quality`	`string`	N/A	The quality by which to filter asset data.
`resolution`	`string`	N/A	The time interval over which to aggregate data.
`count`	`double` or `null`	`null`	The total number of data points for the given variables over the current time interval.
`average`	`double` or `null`	`null`	The mean of the given variables

Field name	Supported types	Default type	Description
			' values over the current time interval.
`min`	`double` or `null`	`null`	The minimum of the given variables ' values over the current time interval.
`max`	`boolean` or `null`	`null`	The maximum of the given variables ' values over the current time interval.
`sum`	`string` or `null`	`null`	The sum of the given variables' values over the current time interval.
`recordVersion`	`long` or `null`	`null`	The version number for the record. You can use the version number to select the latest record. Newer records have larger version numbers.

Example Metric data in the cold tier


```
{"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637334060,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"double":16.0},"min":{"double":1.0},"max":{"double":31.0},"sum":
{"double":496.0},"recordVersion":null}
  {"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637334120,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"double":46.0},"min":{"double":32.0},"max":{"double":60.0},"sum":
{"double":1334.0},"recordVersion":null}
```

```
  {"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637334540,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"double":16.0},"min":{"double":1.0},"max":{"double":31.0},"sum":
{"double":496.0},"recordVersion":null}
  {"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637334600,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"double":46.0},"min":{"double":32.0},"max":{"double":60.0},"sum":
{"double":1334.0},"recordVersion":null}
  {"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637335020,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"double":16.0},"min":{"double":1.0},"max":{"double":31.0},"sum":
{"double":496.0},"recordVersion":null}
```


Asset metadata

When you enable AWS IoT SiteWise to export data to the cold tier for the first time, asset metadata is exported to the cold tier. After the initial configuration, AWS IoT SiteWise exports asset metadata to the tier only when you change asset model definitions or asset definitions. Asset metadata is saved in the cold tier in the newline delimited JSON (`.ndjson`) format.

File path

AWS IoT SiteWise stores asset metadata in the cold tier using the following template.

```
{keyPrefix}/asset_metadata/asset_{assetId}.ndjson
```

Every file path to asset metadata in the cold tier contains the following components.

Path component	Description
`keyPrefix`	The Amazon S3 prefix that you specified in the AWS IoT SiteWises storage configuration. Amazon S3 uses the prefix as a folder name in the bucket.
`asset_metadata`	The folder that stores asset metadata. The `asset_metadata` folder is saved in the prefix folder.

Path component	Description
`fileName`	The file name uses the underscore (_) character as a delimiter to separate the following: • The `asset` prefix. • The `assetId` value. The file is saved in the `.ndjson` format.

Example file path to asset metadata in the colder tier

`keyPrefix/asset_metadata/asset_35901915-d476-4dca-8637-d9ed4df939ed.ndjson`

Fields

The schema of asset metadata that is exported to the cold tier contains the following fields.

Field name	Description
`assetId`	The ID of the asset.
`assetName`	The name of the asset.
`assetExternalId`	The external ID of the asset.
`assetModelId`	The ID of the asset model used to create this asset.
`assetModelName`	The name of the asset model.
`assetModelExternalId`	The external ID of the asset model.
`assetPropertyId`	The ID of the asset property.
`assetPropertyName`	The name of the asset property.
`assetPropertyExternalId`	The external ID of the asset property.

Field name	Description
`assetPropertyDataType`	The data type of the asset property.
`assetPropertyUnit`	The unit of the asset property (for example, `Newtons` and `RPM`).
`assetPropertyAlias`	The alias that identifies the asset property, such as an OPC-UA server data stream path (for example, `/company/windfarm/3/turbine/7/temperature`).
`timeSeriesId`	The ID that identifies the time series data from equipment, metrics, or transforms. You can use this field to join raw data and asset metadata in queries.
`timeSeriesBucket`	A hexadecimal number between 00 and ff. This number is derived from `timeSeriesId` . This partition is used to increase throughput when AWS IoT SiteWise writes to the cold tier. When you use Amazon Athena to run queries, you can use the partition for fine-grain partitioning to improve query performance. `timeSeriesBucket` and `seriesBucket` in the file path to raw data are the same number.
`assetCompositeModelId`	The ID of the composite model.
`assetCompositeModelExternalId`	The external ID of the composite model.
`assetCompositeModelDescription`	The description of the composite model.
`assetCompositeModelName`	The name of the composite model.
`assetCompositeModelType`	The type of the composite model. For alarm composite models, this type is `AWS/ALARM` .

Field name	Description
assetCreationDate	The date the asset was created, in Unix epoch time.
assetLastUpdateDate	The date the asset was last updated, in Unix epoch time.
assetStatusErrorCode	The error code.
assetStatusErrorMessage	The error message.
assetStatusState	The current status of the asset.

Example asset metadata in the cold tier


```
  {"assetId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d","assetExternalId":null,"assetName":"Wind Turbine Asset
 2","assetModelId":"ec1d924f-f07d-444f-b072-
e2994c165d35","assetModelExternalId":null,"assetModelName":"Wind
 Turbine Asset Model","assetPropertyId":"95e63da7-d34e-43e1-
bc6f-1b490154b07a","assetPropertyExternalId":null,"assetPropertyName":"Temperature","assetPrope
Washington/Seattle/WT2/temp","timeSeriesId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeSeriesBucket":"f6","assetArn":null,"assetCompositeModelDescription":nul
  {"assetId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d","assetExternalId":null,"assetName":"Wind Turbine Asset
 2","assetModelId":"ec1d924f-f07d-444f-b072-
e2994c165d35","assetModelExternalId":null,"assetModelName":"Wind Turbine Asset
 Model","assetPropertyId":"c706d54d-4c11-42dc-9a01-63662fc697b4","assetPropertyExternalId":null
Washington/Seattle/WT2/pressure","timeSeriesId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d_c706d54d-4c11-42dc-9a01-63662fc697b4","timeSeriesBucket":"1e","assetArn":null,"ass
  {"assetId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d","assetExternalId":null,"assetName":"Wind Turbine Asset
 2","assetModelId":"ec1d924f-f07d-444f-b072-
e2994c165d35","assetModelExternalId":null,"assetModelName":"Wind
 Turbine Asset Model","assetPropertyId":"8cf1162f-dead-4fbe-b468-
c8e24cde9f50","assetPropertyExternalId":null,"assetPropertyName":"Max
 Temperature","assetPropertyDataType":"DOUBLE","assetPropertyUnit":null,"assetPropertyAlias":nu
```

```
e6db-40fa-9845-ed0dddd4c77d_8cf1162f-dead-4fbe-b468-
c8e24cde9f50","timeSeriesBucket":"d7","assetArn":null,"assetCompositeModelDescription":null,"as

 {"assetId":"3a5f2a22-3b37-4332-9c1c-404ea1d73fab","assetExternalId":null,"assetName":"BatchAss
ebc75e75e827","assetModelExternalId":null,"assetModelName":"FlashTestAssetModelDouble","assetPr
b410-
ab401a9176ed","assetPropertyExternalId":null,"assetPropertyName":"measurementProperty","assetPr
ae89-
ff316f5ff8aa","timeSeriesBucket":"af","assetArn":null,"assetCompositeModelDescription":null,"as
```


Asset hierarchy metadata

When you enable AWS IoT SiteWise to save data the in cold tier for the first time, asset hierarchy metadata is exported to the cold tier. After the initial configuration, AWS IoT SiteWise exports asset hierarchy metadata to the cold tier only when you make changes to asset model or asset definitions. Asset hierarchy metadata is saved in the cold tier in the newline delimited JSON (`.ndjson`) format.

An external identifier for the hierarchy, target asset, or source asset is retrieved by calling the DescribeAsset API.

File path

AWS IoT SiteWise stores asset hierarchy metadata in the cold tier using the following template.

```
{keyPrefix}/asset_hierarchy_metadata/{parentAssetId}_{hierarchyId}.ndjson
```

Every file path to asset hierarchy metadata in the cold tier contains the following components.

Path component	Description
`keyPrefix`	The Amazon S3 prefix that you specified in the AWS IoT SiteWise storage configuration. Amazon S3 uses the prefix as a folder name in the bucket.

Path component	Description
`asset_hierarchy_metadata`	The folder that stores asset hierarchy metadata. The `asset_hierarchy_metadata` folder is saved in the prefix folder.
`fileName`	The file name uses the underscore (_) character as a delimiter to separate the following: • The `parentAssetId` value. • The `hierarchyId` value. The file is saved in the `.ndjson` format.

Example file path to asset hierarchy metadata in the cold tier

```
keyPrefix/asset_hierarchy_metadata/35901915-d476-4dca-8637-
d9ed4df939ed_c5b3ced8-589a-48c7-9998-cdccfc9747a0.ndjson
```

Fields

The schema of asset hierarchy metadata that is exported to the cold tier contains the following fields.

Field name	Description
`sourceAssetId`	The ID of the source asset in this asset relationship.
`targetAssetId`	The ID of the target asset in this asset relationship.
`hierarchyId`	The ID of the hierarchy.
`associationType`	The association type of this asset relationship.

Field name	Description
	The value must be CHILD. The target asset is a child asset of the source asset.

Example asset hierarchy metadata in the cold tier

```
{"sourceAssetId":"80388e72-2284-44fb-9c89-
bfbaf0dfedd2","targetAssetId":"2b866c25-0c74-4750-bdf5-
b73683c8a2a2","hierarchyId":"bbed9f59-0412-4585-
a61d-6044db526aee","associationType":"CHILD"}
  {"sourceAssetId":"80388e72-2284-44fb-9c89-
bfbaf0dfedd2","targetAssetId":"6b51246e-984d-460d-
bc0b-470ea47d1e31","hierarchyId":"bbed9f59-0412-4585-
a61d-6044db526aee","associationType":"CHILD"}
```

To view your data in the cold tier

1. Navigate to the Amazon S3 console.
2. In the navigation pane, choose **Buckets**, and then choose your Amazon S3 bucket.
3. Navigate to the folder that contains the raw data, asset metadata, or asset hierarchy metadata.
4. Select the files, and then from **Actions**, choose **Download**.

Storage data index files

AWS IoT SiteWise uses these files to optimize data query performance. They appear in your Amazon S3 bucket, but you don't need to use them.

File path

AWS IoT SiteWise stores data index files in the cold tier using the following template.

```
keyPrefix/index/series=timeseriesId/startYear=startYear/startMonth=startMonth/
startDay=startDay/index_timeseriesId_startTimestamp_quality
```

Example file path to data storage index file

```
keyPrefix/index/series=7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-
d34e-43e1-bc6f-1b490154b07a/startYear=2022/startMonth=02/startDay=03/
```

```
index_7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a_1643846400_GOOD
```

Security in AWS IoT SiteWise

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs. To learn about the compliance programs that apply to AWS IoT SiteWise, see AWS services in scope by compliance program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS IoT SiteWise. The following topics show you how to configure AWS IoT SiteWise to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS IoT SiteWise resources.

Topics

Data protection in AWS IoT SiteWise

The AWS shared responsibility model applies to data protection in AWS IoT SiteWise. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with AWS IoT SiteWise or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Topics

Internetwork traffic privacy

Connections between AWS IoT SiteWise and on-premises applications, such as SiteWise Edge gateways, are secured over Transport Layer Security (TLS) connections. For more information, see Encryption in transit.

AWS IoT SiteWise doesn't support connections between Availability Zones within an AWS Region or connections between AWS accounts.

You can configure IAM Identity Center in only one Region at a time. SiteWise Monitor connects to the Region that you configured for IAM Identity Center. This means that you use one Region for IAM Identity Center access, but you can create portals in any Region.

Data encryption

Data encryption refers to protecting data while in-transit (as it travels to and from AWS IoT SiteWise, and between SiteWise Edge gateways and servers), and at rest (while it is stored on local devices or in AWS services). You can protect data in transit using Transport Layer Security (TLS) or at rest using client-side encryption.

> **Note**
>
> AWS IoT SiteWise edge processing exposes APIs that are hosted within SiteWise Edge gateways and accessible over the local network. These APIs are exposed over a TLS connection backed by a server-certificate owned by the AWS IoT SiteWise Edge connector. For client authentication, these APIs use an access-control password. The server-certificate private-key and the access-control password are both stored on disk. AWS IoT SiteWise edge processing relies on file-system encryption for the security of these credentials at rest.

For more information about server-side encryption and client-side encryption, review the topics listed below.

Topics

- Encryption at rest
- Encryption in transit
- Key management

Encryption at rest

AWS IoT SiteWise stores your data in the AWS Cloud and on AWS IoT SiteWise Edge gateways.

Data at rest in the AWS Cloud

AWS IoT SiteWise stores data in other AWS services that encrypt data at rest by default. Encryption at rest integrates with AWS Key Management Service (AWS KMS) for managing the encryption key that is used to encrypt your asset property values and aggregate values in AWS IoT SiteWise. You can choose to use a customer managed key to encrypt asset property values and aggregate values in AWS IoT SiteWise. You can create, manage, and view your encryption key through AWS KMS.

You can choose an AWS owned key to encrypt your data, or choose a customer managed keyto encrypt your asset property values and aggregate values:

How it works

Encryption at rest integrates with AWS KMS for managing the encryption key that is used to encrypt your data.

- AWS owned key – Default encryption key. AWS IoT SiteWise owns this key. You can't view this key in your AWS account. You also can't see operations on the key in AWS CloudTrail logs. You can use this key at no additional charge.
- Customer managed key – The key is stored in your account, which you create, own, and manage. You have full control over the KMS key. Additional AWS KMS charges apply.

AWS owned keys

AWS owned keys aren't stored in your account. They're part of a collection of KMS keys that AWS owns and manages for use in multiple AWS accounts. AWS services can use AWS owned keys to protect your data.

You can't view, manage, use AWS owned keys, or audit their use. However, you don't need to do any work or change any programs to protect the keys that encrypt your data.

You're not charged a monthly fee or a usage fee if you use AWS owned keys, and they don't count against AWS KMS quotas for your account.

Customer managed keys

Customer managed keys are KMS keys in your account that you create, own, and manage. You have full control over these KMS keys, such as the following:

- Establishing and maintaining their key policies, IAM policies, and grants
- Enabling and disabling them
- Rotating their cryptographic material
- Adding tags
- Creating aliases that refer to them
- Scheduling them for deletion

You can also use CloudTrail and Amazon CloudWatch Logs to track the requests that AWS IoT SiteWise sends to AWS KMS on your behalf.

If you're using customer managed keys, you need to grant AWS IoT SiteWise access to the KMS key stored in your account. AWS IoT SiteWise uses envelope encryption and key hierarchy to encrypt data. Your AWS KMS encryption key is used to encrypt the root key of this key hierarchy. For more information, see Envelope encryption in the *AWS Key Management Service Developer Guide*.

The following example policy grants AWS IoT SiteWise permissions to a create customer managed key on your behalf. When you create your key, you need to allow the `kms:CreateGrant` and `kms:DescribeKey` actions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1603902045292",
      "Action": [
        "kms:CreateGrant",
        "kms:DescribeKey"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

The encryption context for your created grant uses your `aws:iotsitewise:subscriberId` and account ID.

Data at rest on SiteWise Edge gateways

AWS IoT SiteWise gateways store the following data on the local file system:

- OPC-UA source configuration information
- The set of OPC-UA data stream paths from connected OPC-UA sources
- Industrial data cached when the SiteWise Edge gateway loses connection to the internet

SiteWise Edge gateways run on AWS IoT Greengrass. AWS IoT Greengrass relies on Unix file permissions and full-disk encryption (if enabled) to protect data at rest on the core. It's your responsibility to secure the file system and device.

However, AWS IoT Greengrass does encrypt local copies of your OPC-UA server secrets retrieved from Secrets Manager. For more information, see Secrets encryption in the *AWS IoT Greengrass Version 1 Developer Guide*.

For more information about encryption at rest on AWS IoT Greengrass cores, see Encryption at rest in the *AWS IoT Greengrass Version 1 Developer Guide*.

Encryption in transit

AWS IoT SiteWise has three modes of communication where data is in transit:

- Over the internet – Communication between local devices (including SiteWise Edge gateways) and AWS IoT SiteWise is encrypted.
- Over the local network – Communication between OpsHub for SiteWise application and SiteWise Edge gateways is always encrypted. Communication between the SiteWise monitor application running within your browser and SiteWise Edge gateways is always encrypted. Communication between SiteWise Edge gateways and OPC-UA sources can be encrypted.
- Between components on SiteWise Edge gateways – Communication between AWS IoT Greengrass components on SiteWise Edge gateways isn't encrypted.

Topics

- Data in transit over the internet

- Data in transit over the local network
- Data in transit between local components on SiteWise Edge gateways

Data in transit over the internet

AWS IoT SiteWise uses Transport Layer Security (TLS) to encrypt all communication over the internet. All data sent to the AWS Cloud is sent over a TLS connection using MQTT or HTTPS protocols, so it's secure by default. SiteWise Edge gateways, which run on AWS IoT Greengrass, and property value notifications use the AWS IoT transport security model. For more information, see Transport security in the *AWS IoT Developer Guide*.

Data in transit over the local network

SiteWise Edge gateways follow OPC-UA specifications for communication with local OPC-UA sources. It's your responsibility to configure your sources to use a message security mode that encrypts data in transit.

If you choose a *sign* message security mode, data in transit between SiteWise Edge gateways and sources is signed but not encrypted. If you choose a *sign and encrypt* message security mode, the data in transit between SiteWise Edge gateways and sources is signed and encrypted. For more information about configuring sources, see Configuring data sources.

The communication between the edge console application and SiteWise Edge gateways is always encrypted by TLS. The SiteWise Edge connector on the SiteWise Edge gateway generates and stores a self-signed certificate to be able to establish a TLS connection with the edge console for AWS IoT SiteWise application. You will need to copy this certificate from your SiteWise Edge gateway to the edge console for AWS IoT SiteWise application before you connect the application to the SiteWise Edge gateway. This ensures that the edge console for AWS IoT SiteWise application is able to verify that it has connected to your trusted SiteWise Edge gateway.

In addition to TLS for secrecy and server authenticity, SiteWise Edge uses the SigV4 protocol to establish the authenticity of the edge console application. The SiteWise Edge connector on the SiteWise Edge gateway accepts and stores a password to be able to verify incoming connections from the edge console application, SiteWise Monitor application running within browsers, and other clients based on the AWS IoT SiteWise SDK.

For more information about generating the password and server certificate, see the section called “Managing SiteWise Edge gateways”.

Data in transit between local components on SiteWise Edge gateways

SiteWise Edge gateways run on AWS IoT Greengrass, which doesn't encrypt data exchanged locally on the AWS IoT Greengrass core because the data doesn't leave the device. This includes communication between AWS IoT Greengrass components such as the AWS IoT SiteWise connector. For more information, see Data on the core device in the *AWS IoT Greengrass Version 1 Developer Guide*.

Key management

AWS IoT SiteWise cloud key management

By default, AWS IoT SiteWise uses AWS managed keys to protect your data in the AWS Cloud. You can update your settings to use a customer managed key to encrypt some data in AWS IoT SiteWise. You can create, manage, and view your encryption key through AWS Key Management Service (AWS KMS).

AWS IoT SiteWise supports server-side encryption with customer managed keys stored in AWS KMS to encrypt the following data:

- Asset property values
- Aggregate values

> **Note**
>
> Other data and resources are encrypted using the default encryption with keys managed by AWS IoT SiteWise. This key is stored in the AWS IoT SiteWise account.

For more information, see What is AWS Key Management Service? in the *AWS Key Management Service Developer Guide*.

Enable encryption using customer managed keys

To use customer managed keys with AWS IoT SiteWise, you need to update your AWS IoT SiteWise settings.

To enable encryption using KMS keys

1.

Navigate to the AWS IoT SiteWise console.

2. Choose **Account Settings** and choose **Edit** to open the **Edit account settings** page.
3. For **Encryption key type**, choose **Choose a different AWS KMS key**. This enables encryption with customer managed keys stored in AWS KMS.

> **Note**
>
> Currently, you can only use customer managed key encryption for asset property values and aggregate values.

4. Choose your KMS key with one of the following options:

   - **To use an existing KMS key** – Choose your KMS key alias from the list.
   - **To create a new KMS key** – Choose **Create an AWS KMS key**.

> **Note**
>
> This opens the AWS KMS dashboard. For more information about creating a KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

5. Choose **Save** to update your settings.

SiteWise Edge gateway key management

SiteWise Edge gateways run on AWS IoT Greengrass, and AWS IoT Greengrass core devices use public and private keys to authenticate with the AWS Cloud and encrypt local secrets, such as OPC-UA authentication secrets. For more information, see Key management in the *AWS IoT Greengrass Version 1 Developer Guide*.

Identity and access management for AWS IoT SiteWise

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS IoT SiteWise resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in AWS IoT SiteWise.

Service user – If you use the AWS IoT SiteWise service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS IoT SiteWise features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in AWS IoT SiteWise, see Troubleshooting AWS IoT SiteWise identity and access.

Service administrator – If you're in charge of AWS IoT SiteWise resources at your company, you probably have full access to AWS IoT SiteWise. It's your job to determine which AWS IoT SiteWise features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with AWS IoT SiteWise, see How AWS IoT SiteWise works with IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS IoT SiteWise. To view example AWS IoT SiteWise identity-based policies that you can use in IAM, see AWS IoT SiteWise identity-based policy examples.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center User Guide* and Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.

- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

How AWS IoT SiteWise works with IAM

Before you use AWS Identity and Access Management (IAM) to manage access to AWS IoT SiteWise, you should understand what IAM features are available to use with AWS IoT SiteWise.

IAM feature	Suppor by AWS IoT SiteWis
Identity-based policies with resource-level permissions	Yes
Policy actions	Yes
Policy resources	Yes
Policy condition keys	Yes
Resource-based policies	No
Access control lists (ACLs)	No
Tags-based authorization (ABAC)	Yes
Temporary credentials	Yes
Forward access sessions (FAS)	Yes
Service-linked roles	Yes
Service roles	Yes

To get a high-level view of how AWS IoT SiteWise and other AWS services work with IAM, see AWS services that work with IAM in the *IAM User Guide*.

Contents

AWS IoT SiteWise IAM roles

An IAM role is an entity within your AWS account that has specific permissions.

Using temporary credentials with AWS IoT SiteWise

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

AWS IoT SiteWise supports using temporary credentials.

SiteWise Monitor supports federated users to access portals. Portal users authenticate with their IAM Identity Center or IAM credentials.

> **Important**
>
> Users or roles must have the `iotsitewise:DescribePortal` permission to sign in to the portal.

When a user signs in to a portal, SiteWise Monitor generates a session policy that provides the following permissions:

- Read-only access to the assets and asset data in AWS IoT SiteWise in your account to which that portal's role provides access.
- Access to projects in that portal to which the user has administrator (project owner) or read-only (project viewer) access.

For more information about federated portal user permissions, see Using service roles for AWS IoT SiteWise Monitor.

Forward access sessions (FAS) for AWS IoT SiteWise

Supports forward access sessions (FAS)	Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to

complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

Service-linked roles

Service-linked roles allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view but not edit the permissions for service-linked roles.

AWS IoT SiteWise supports service-linked roles. For details about creating or managing AWS IoT SiteWise service-linked roles, see Using service-linked roles for AWS IoT SiteWise.

Service roles

This feature allows a service to assume a service role on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your AWS account and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, doing so might break the functionality of the service.

AWS IoT SiteWise uses a service role to allow SiteWise Monitor portal users to access some of your AWS IoT SiteWise resources on your behalf. For more information, see Using service roles for AWS IoT SiteWise Monitor.

You must have required permissions before you can create AWS IoT Events alarm models in AWS IoT SiteWise. For more information, see Setting up permissions for AWS IoT Events alarms.

Choosing an IAM role in AWS IoT SiteWise

When you create a `portal` resource in AWS IoT SiteWise, you must choose a role to allow the federated users of your SiteWise Monitor portal to access AWS IoT SiteWise on your behalf. If you have previously created a service role, then AWS IoT SiteWise provides you with a list of roles to choose from. Otherwise, you can create a role with the required permissions when you create a portal. It's important to choose a role that allows access to your assets and asset data. For more information, see Using service roles for AWS IoT SiteWise Monitor.

Authorization based on AWS IoT SiteWise tags

You can attach tags to AWS IoT SiteWise resources or pass tags in a request to AWS IoT SiteWise. To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, or `aws:TagKeys`

condition keys. For more information about tagging AWS IoT SiteWise resources, see Tagging your AWS IoT SiteWise resources.

To view an example identity-based policy for limiting access to a resource based on the tags on that resource, see Viewing AWS IoT SiteWise assets based on tags.

AWS IoT SiteWise identity-based policies

IAM policies let you control who can do what in AWS IoT SiteWise. You can decide what actions are allowed or not and set specific conditions for these actions. For example, you can make rules about who can see or change information in AWS IoT SiteWise. AWS IoT SiteWise supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Policy actions

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

Policy actions in AWS IoT SiteWise use the following prefix before the action: `iotsitewise:`. For example, to grant someone permission to upload asset property data to AWS IoT SiteWise with the `BatchPutAssetPropertyValue` API operation, you include the `iotsitewise:BatchPutAssetPropertyValue` action in their policy. Policy statements must include either an `Action` or `NotAction` element. AWS IoT SiteWise defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows.

```
"Action": [
  "iotsitewise:action1",
  "iotsitewise:action2"
]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action.

```
"Action": "iotsitewise:Describe*"
```

To see a list of AWS IoT SiteWise actions, see Actions Defined by AWS IoT SiteWise in the *IAM User Guide*.

BatchPutAssetPropertyValue authorization

AWS IoT SiteWise authorizes access to the BatchPutAssetPropertyValue action in an unusual way. For most actions, when you allow or deny access, that action returns an error if permissions aren't granted. With `BatchPutAssetPropertyValue`, you can send multiple data entries to different assets and asset properties in a single API request. AWS IoT SiteWise authorizes each data entry independently. For any individual entry that fails authorization in the request, AWS IoT SiteWise includes an `AccessDeniedException` in the returned list of errors. AWS IoT SiteWise receives the data for any entry that authorizes and succeeds, even if another entry in the same request fails.

Important

Before you ingest data to a data stream, do the following:

- Authorize the `time-series` resource if you use a property alias to identify the data stream.
- Authorize the `asset` resource if you use an asset ID to identify the asset that contains the associated asset property.

Policy resources

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

Each IAM policy statement applies to the resources that you specify using their ARNs. An ARN has the following general syntax.

```
arn:${Partition}:${Service}:${Region}:${Account}:${ResourceType}/${ResourcePath}
```

For more information about the format of ARNs, see Amazon Resource Names (ARNs) and AWS service namespaces.

For example, to specify the asset with ID `a1b2c3d4-5678-90ab-cdef-22222EXAMPLE` in your statement, use the following ARN.;

```
"Resource": "arn:aws:iotsitewise:region:123456789012:asset/a1b2c3d4-5678-90ab-
cdef-22222EXAMPLE"
```

To specify all data streams that belong to a specific account, use the wildcard (*):

```
"Resource": "arn:aws:iotsitewise:region:123456789012:time-series/*"
```

To specify all assets that belong to a specific account, use the wildcard (*):

```
"Resource": "arn:aws:iotsitewise:region:123456789012:asset/*"
```

Some AWS IoT SiteWise actions, such as those for creating resources, can't be performed on a specific resource. In those cases, you must use the wildcard (*).

```
"Resource": "*"
```

To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": [
  "resource1",
  "resource2"
```

```
]
```

To see a list of AWS IoT SiteWise resource types and their ARNs, see Resources Defined by AWS IoT SiteWise in the *IAM User Guide*. To learn with which actions you can specify the ARN of each resource, see Actions Defined by AWS IoT SiteWise.

Policy condition keys

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

> **Important**
>
> Many condition keys are specific to a resource, and some API actions use multiple resources. If you write a policy statement with a condition key, use the `Resource` element of the statement to specify the resource to which the condition key applies. If you don't do so, the policy might prevent users from performing the action at all, because the condition check fails for the resources to which the condition key doesn't apply. If you don't want to specify a resource, or if you've written the `Action` element of your policy to include multiple API actions, then you must use the `...IfExists` condition type to ensure that the condition key is ignored for resources that don't use it. For more information, see ...IfExists conditions in the *IAM User Guide*.

AWS IoT SiteWise defines its own set of condition keys and also supports using some global condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

AWS IoT SiteWise condition keys

Condition key	Description	Types
`iotsitewise:isAssociatedWithAssetProperty`	Whether data streams are associated with an asset property. Use this condition key to define permissions based on the existence of an associated asset property for data streams. Example value: `true`	String
`iotsitewise:assetHierarchyPath`	The asset's hierarchy path, which is a string of asset IDs each separated by a forward slash. Use this condition key to define permissions based on a subset of your hierarchy of all assets in your account. Example value: `/a1b2c3d4-5678-90ab-cdef-222222EXAMPLE/a1b2c3d4-5678-90ab-cdef-666666EXAMPLE`	String
`iotsitewise:propertyId`	The ID of an asset property. Use this condition key to define permissions based on a specified property of an asset model. This condition key applies to all assets of that model.	String

Condition key	Description	Types
	Example value: `a1b2c3d4-5678-90ab-cdef-33333EXAMPLE`	
`iotsitewise:childAssetId`	The ID of an asset being associated as a child to another asset. Use this condition key to define permissions based on child assets. To define permissions based on parent assets, use the resource section of a policy statement. Example value: `a1b2c3d4-5678-90ab-cdef-666666EXAMPLE`	String
`iotsitewise:iam`	The ARN of an IAM identity when listing access policies. Use this condition key to define access policy permissions for an IAM identity. Example value: `arn:aws:iam::123456789012:user/JohnDoe`	String, Null
`iotsitewise:propertyAlias`	The alias that identifies an asset property or data stream. Use this condition key to define permissions based on the alias.	String

Condition key	Description	Types
`iotsitewise:user`	The ID of an IAM Identity Center user when listing access policies. Use this condition key to define access policy permissions for an IAM Identity Center user. Example value: `a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE`	String, Null
`iotsitewise:group`	The ID of an IAM Identity Center group when listing access policies. Use this condition key to define access policy permissions for an IAM Identity Center group. Example value: `a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-bbbbbEXAMPLE`	String, Null
`iotsitewise:portal`	The ID of a portal in an access policy. Use this condition key to define access policy permissions based on a portal. Example value: `a1b2c3d4-5678-90ab-cdef-77777EXAMPLE`	String, Null

Condition key	Description	Types
`iotsitewise:project`	The ID of a project in an access policy, or the ID of a project for a dashboard . Use this condition key to define dashboard or access policy permissions based on a project. Example value: `a1b2c3d4-5678-90ab-cdef-888 88EXAMPLE`	String, Null

To learn with which actions and resources you can use a condition key, see Actions Defined by AWS IoT SiteWise.

Examples

To view examples of AWS IoT SiteWise identity-based policies, see AWS IoT SiteWise identity-based policy examples.

AWS IoT SiteWise identity-based policy examples

By default, entities (users and roles) don't have permission to create or modify AWS IoT SiteWise resources. They also can't perform tasks using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To adjust permissions, an AWS Identity and Access Management (IAM) administrator must do the following:

1. Create IAM policies that grant users and roles permission to perform specific API operations on resources they need.
2. Attach those policies to the users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating policies on the JSON tab in the *IAM User Guide*.

Topics

- Policy best practices
- Using the AWS IoT SiteWise console
- Allowing users to view their own permissions
- Allowing users to ingest data to assets in one hierarchy
- Viewing AWS IoT SiteWise assets based on tags

Policy best practices

Identity-based policies determine whether someone can create, access, or delete AWS IoT SiteWise resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.

- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the AWS IoT SiteWise console

To access the AWS IoT SiteWise console, you need a basic set of permissions. These permissions let you see and manage details about the AWS IoT SiteWise resources in your AWS account.

If you make a policy that's too restrictive, the console might not work as expected for users or roles (entities) with that policy. To ensure that those entities can still use the AWS IoT SiteWise console, attach the AWSIoTSiteWiseConsoleFullAccess managed policy to them or define equivalent permissions for those entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

If entities are only using the AWS Command Line Interface (CLI) or the AWS IoT SiteWise API, and not the console, they don't need these minimum permissions. In that case, just give them access to the specific actions they need for their API tasks.

Allowing users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
```

```
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Allowing users to ingest data to assets in one hierarchy

In this example, you want to grant a user in your AWS account access to write data to all asset properties in a specific hierarchy of assets, starting from the root asset `a1b2c3d4-5678-90ab-cdef-22222EXAMPLE`. The policy grants the `iotsitewise:BatchPutAssetPropertyValue` permission to the user. This policy uses the `iotsitewise:assetHierarchyPath` condition key to restrict access to assets whose hierarchy path matches the asset or its descendants.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PutAssetPropertyValuesForHierarchy",
      "Effect": "Allow",
      "Action": "iotsitewise:BatchPutAssetPropertyValue",
      "Resource": "arn:aws:iotsitewise:*:*:asset/*",
      "Condition": {
        "StringLike": {
          "iotsitewise:assetHierarchyPath": [
            "/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
            "/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/*"
```

```
          ]
        }
      }
    }
  ]
}
```

Viewing AWS IoT SiteWise assets based on tags

Use conditions in your identity-based policy to control access to AWS IoT SiteWise resources based on tags. This example shows how to create a policy that allows asset viewing. However, permission is granted only if the asset tag `Owner` has the value of that user's user name. This policy also grants permission to complete this action on the console.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListAllAssets",
      "Effect": "Allow",
      "Action": [
        "iotsitewise:ListAssets",
        "iotsitewise:ListAssociatedAssets"
      ],
      "Resource": "*"
    },
    {
      "Sid": "DescribeAssetIfOwner",
      "Effect": "Allow",
      "Action": "iotsitewise:DescribeAsset",
      "Resource": "arn:aws:iotsitewise:*:*:asset/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Owner": "${aws:username}"
        }
      }
    }
  ]
}
```

Attach this policy to the users in your account. If a user named `richard-roe` attempts to view an AWS IoT SiteWise asset, the asset must be tagged `Owner=richard-roe` or `owner=richard-roe`.

Otherwise, Richard is denied access. The condition tag key names are not case-sensitive. So, `Owner` matches both `Owner` and `owner`. For more information, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including

each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.

- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

AWS managed policies for AWS IoT SiteWise

Simplify adding permissions to users, groups, and roles using AWS managed policies rather than to writing policies yourself. It takes time and expertise to create IAM customer managed policies that provide your team precise permissions. For a faster setup, consider using our AWS managed policies for common use cases. Find AWS managed policies in your AWS account. For more information about AWS managed policies, see AWS managed policies in the *IAM User Guide*.

AWS services take care of updating and maintaining AWS managed policies, meaning you cannot modify these policies' permissions. Occasionally, AWS IoT SiteWise may add permissions to accommodate new features, impacting all identities with the policy attached. Such updates are common with the introduction of new services or features. However, permissions are never removed, ensuring your setups remain intact.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the **ReadOnlyAccess** AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list with descriptions of job function policies, see AWS managed policies for job functions in the *IAM User Guide*.

AWS managed policy: AWSIoTSiteWiseReadOnlyAccess

Use the `AWSIoTSiteWiseReadOnlyAccess` AWS managed policy to allow read-only access to AWS IoT SiteWise.

You can attach the AWSIoTSiteWiseReadOnlyAccess policy to your IAM identities.

Service-level permissions

This policy provides read-only access to AWS IoT SiteWise. No other service permissions are included in this policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:Describe*",
                "iotsitewise:List*",
                "iotsitewise:BatchGet*",
                "iotsitewise:Get*"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS managed policy: AWSServiceRoleForIoTSiteWise

The AWSServiceRoleForIoTSiteWise role uses the AWSServiceRoleForIoTSiteWise policy with the following permissions. This policy:

- Allows AWS IoT SiteWise to deploy SiteWise Edge gateways (which run on AWS IoT Greengrass).
- Allows AWS IoT SiteWise to perform logging.
- Allows AWS IoT SiteWise to run a metadata search query, against the AWS IoT TwinMaker database.

If you are using AWS IoT SiteWise with a singe user account,the AWSServiceRoleForIoTSiteWise role creates the AWSServiceRoleForIoTSiteWise policy in your IAM account, and attaches it to the AWSServiceRoleForIoTSiteWise service-linked roles for AWS IoT SiteWise.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowSiteWiseReadGreenGrass",
   "Effect": "Allow",
   "Action": [
    "greengrass:GetAssociatedRole",
    "greengrass:GetCoreDefinition",
    "greengrass:GetCoreDefinitionVersion",
    "greengrass:GetGroup",
    "greengrass:GetGroupVersion"
   ],
   "Resource": "*"
  },
  {
   "Sid": "AllowSiteWiseAccessLogGroup",
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogGroup",
    "logs:DescribeLogGroups"
   ],
   "Resource": "arn:aws:logs:*:*:log-group:/aws/iotsitewise*"
  },
  {
   "Sid": "AllowSiteWiseAccessLog",
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogStream",
    "logs:DescribeLogStreams",
    "logs:PutLogEvents"
   ],
   "Resource": "arn:aws:logs:*:*:log-group:/aws/iotsitewise*:log-stream:*"
  },
  {
   "Sid": "AllowSiteWiseAccessSiteWiseManagedWorkspaceInTwinMaker",
   "Effect": "Allow",
   "Action": [
    "iottwinmaker:GetWorkspace",
    "iottwinmaker:ExecuteQuery"
   ],
   "Resource": "arn:aws:iottwinmaker:*:*:workspace/*",
   "Condition": {
    "ForAnyValue:StringEquals": {
```

```
      "iottwinmaker:linkedServices": [
       "IOTSITEWISE"
      ]
     }
    }
   }
  ]
}
```

AWS IoT SiteWise updates to AWS managed policies

You can view details about updates to AWS managed policies for AWS IoT SiteWise, beginning from when this service began tracking the changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the AWS IoT SiteWise Document history page.

Change	Description	Date
AWSServiceRoleForIoTSiteWise – Update to an existing policy	AWS IoT SiteWise now can run a metadata search query, against the AWS IoT TwinMaker database.	November 6, 2023
AWSIoTSiteWiseReadOnlyAccess – Update to an existing policy	AWS IoT SiteWise added a new policy prefix, `BatchGet*` , that enables you to do batch read operations.	September 16, 2022
AWSIoTSiteWiseReadOnlyAccess – New policy	AWS IoT SiteWise added a new policy to grant read-only access to AWS IoT SiteWise.	November 24, 2021
AWS IoT SiteWise started tracking changes	AWS IoT SiteWise started tracking changes for its AWS managed policies.	November 24, 2021

Using service-linked roles for AWS IoT SiteWise

AWS IoT SiteWise uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to AWS IoT SiteWise. Service-linked

roles are predefined by AWS IoT SiteWise and include all the permissions that the service requires to call other AWS services on your behalf.

Service-linked roles simplify the configuration of AWS IoT SiteWise by automatically including all necessary permissions. AWS IoT SiteWise defines the permissions of its service-linked roles, and unless defined otherwise, only AWS IoT SiteWise can assume its roles. The defined permissions include the trust policy and the permissions policy. And that permissions policy can't be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This protects your AWS IoT SiteWise resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS services that work with IAM and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Topics

Service-linked role permissions for AWS IoT SiteWise

AWS IoT SiteWise uses the service-linked role named **AWSServiceRoleForIoTSiteWise**. AWS IoT SiteWise uses this service-linked role to deploy SiteWise Edge gateways (which run on AWS IoT Greengrass) and perform logging.

The `AWSServiceRoleForIoTSiteWise` service-linked role uses the `AWSServiceRoleForIoTSiteWise` policy with the following permissions. This policy:

- Allows AWS IoT SiteWise to deploy SiteWise Edge gateways (which run on `AWS IoT Greengrass`).
- Allows AWS IoT SiteWise to perform logging.

- Allows AWS IoT SiteWise to run a metadata search query, against the AWS IoT TwinMaker database.

For more information on the allowed actions in `AWSServiceRoleForIoTSiteWise`, see AWS managed policies for AWS IoT SiteWise.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowSiteWiseReadGreenGrass",
   "Effect": "Allow",
   "Action": [
    "greengrass:GetAssociatedRole",
    "greengrass:GetCoreDefinition",
    "greengrass:GetCoreDefinitionVersion",
    "greengrass:GetGroup",
    "greengrass:GetGroupVersion"
   ],
   "Resource": "*"
  },
  {
   "Sid": "AllowSiteWiseAccessLogGroup",
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogGroup",
    "logs:DescribeLogGroups"
   ],
   "Resource": "arn:aws:logs:*:*:log-group:/aws/iotsitewise*"
  },
  {
   "Sid": "AllowSiteWiseAccessLog",
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogStream",
    "logs:DescribeLogStreams",
    "logs:PutLogEvents"
   ],
   "Resource": "arn:aws:logs:*:*:log-group:/aws/iotsitewise*:log-stream:*"
  },
  {
   "Sid": "AllowSiteWiseAccessSiteWiseManagedWorkspaceInTwinMaker",
```

```
    "Effect": "Allow",
    "Action": [
     "iottwinmaker:GetWorkspace",
     "iottwinmaker:ExecuteQuery"
    ],
    "Resource": "arn:aws:iottwinmaker:*:*:workspace/*",
    "Condition": {
     "ForAnyValue:StringEquals": {
      "iottwinmaker:linkedServices": [
       "IOTSITEWISE"
      ]
     }
    }
   }
  ]
}
```

You can use the logs to monitor and troubleshoot your SiteWise Edge gateways. For more information, see Monitoring SiteWise Edge gateway logs.

To allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role, first configure permissions. For more information, see Service-linked role permissions in the *IAM User Guide*.

Creating a service-linked role for AWS IoT SiteWise

You don't need to manually create a service-linked role. When you perform the following operations in the AWS IoT SiteWise console, AWS IoT SiteWise creates the service-linked role for you.

- Create a Greengrass V1 gateway.
- Configure the logging option.
- Choosing the opt-in button in the execute query banner.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you perform any operation in the AWS IoT SiteWise console, AWS IoT SiteWise creates the service-linked role for you again.

You can also use the IAM console or API to create a service-linked role for AWS IoT SiteWise.

- To do so in the IAM console, create a role with the **AWSServiceRoleForIoTSiteWise** policy and a trust relationship with `iotsitewise.amazonaws.com`.
- To do so using the AWS CLI or IAM API, create a role with the `arn:aws:iam::aws:policy/aws-service-role/AWSServiceRoleForIoTSiteWise` policy and a trust relationship with `iotsitewise.amazonaws.com`.

For more information, see Creating a service-linked role in the *IAM User Guide*.

If you delete this service-linked role, you can use this same process to create the role again.

Editing a service-linked role for AWS IoT SiteWise

AWS IoT SiteWise doesn't allow you to edit the AWSServiceRoleForIoTSiteWise service-linked role. After you create a service-linked role, you can't change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Deleting a service-linked role for AWS IoT SiteWise

If a feature or service requiring a service-linked role is no longer in use, it's advisable to delete the associated role. This is to avoid having an inactive entity that isn't being monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

> **Note**
>
> If the AWS IoT SiteWise service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try again.

To delete AWS IoT SiteWise resources used by the AWSServiceRoleForIoTSiteWise

1. Disable logging for AWS IoT SiteWise. For more information, see Changing your logging level
2. Delete any active SiteWise Edge gateways.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForIoTSiteWise service-linked role. For more information, see Deleting a Service-Linked Role in the *IAM User Guide*.

Supported Regions for AWS IoT SiteWise service-linked roles

AWS IoT SiteWise supports using service-linked roles in all of the Regions where the service is available. For more information, see AWS IoT SiteWise Endpoints and Quotas.

Using service roles for AWS IoT SiteWise Monitor

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

To allow federated SiteWise Monitor portal users to access your AWS IoT SiteWise and AWS IAM Identity Center resources, you must attach a service role to each portal that you create. The service role must specify SiteWise Monitor as a trusted entity and include the AWSIoTSiteWiseMonitorPortalAccess managed policy or define equivalent permissions. This policy is maintained by AWS and defines the set of permissions that SiteWise Monitor uses to access your AWS IoT SiteWise and IAM Identity Center resources.

When you create a SiteWise Monitor portal, you must choose a role that allows users of that portal to access your AWS IoT SiteWise and IAM Identity Center resources. The AWS IoT SiteWise console can create and configure the role for you. You can edit the role in IAM later. Your portal users will have issues using their SiteWise Monitor portals if you remove the required permissions from the role or delete the role.

> **Note**
>
> Portals created before April 29, 2020 didn't require service roles. If you created portals before this date, you must attach service roles to continue using them. To do so, navigate to the **Portals** page in the AWS IoT SiteWise console, and then choose **Migrate all portals to use IAM roles**.

The following sections describe how to create and manage the SiteWise Monitor service role in the AWS Management Console or the AWS Command Line Interface.

Contents

- Creating a SiteWise Monitor service role (AWS IoT SiteWise console)
- Creating a SiteWise Monitor service role (IAM console)
- Changing a portal's service role (console)
- Managing the SiteWise Monitor service role (CLI)
  - Finding a portal's service role (CLI)
  - Creating the SiteWise Monitor service role (CLI)
- SiteWise Monitor updates to AWSIoTSiteWiseMonitorServiceRole

Service role permissions for SiteWise Monitor

When you create a portal, AWS IoT SiteWise lets you create a role whose name starts with **AWSIoTSiteWiseMonitorServiceRole**. This role allows federated SiteWise Monitor users to access your portal configuration, assets, asset data, and IAM Identity Center configuration.

The role trusts the following service to assume the role:

- `monitor.iotsitewise.amazonaws.com`

The role uses the following permissions policy, whose name starts with **AWSIoTSiteWiseMonitorServicePortalPolicy**, to allow SiteWise Monitor users to complete actions on resources in your account. The AWSIoTSiteWiseMonitorPortalAccess managed policy defines equivalent permissions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:DescribePortal",
                "iotsitewise:CreateProject",
                "iotsitewise:DescribeProject",
                "iotsitewise:UpdateProject",
                "iotsitewise:DeleteProject",
                "iotsitewise:ListProjects",
                "iotsitewise:BatchAssociateProjectAssets",
                "iotsitewise:BatchDisassociateProjectAssets",
                "iotsitewise:ListProjectAssets",
```

```
                "iotsitewise:CreateDashboard",
                "iotsitewise:DescribeDashboard",
                "iotsitewise:UpdateDashboard",
                "iotsitewise:DeleteDashboard",
                "iotsitewise:ListDashboards",
                "iotsitewise:CreateAccessPolicy",
                "iotsitewise:DescribeAccessPolicy",
                "iotsitewise:UpdateAccessPolicy",
                "iotsitewise:DeleteAccessPolicy",
                "iotsitewise:ListAccessPolicies",
                "iotsitewise:DescribeAsset",
                "iotsitewise:ListAssets",
                "iotsitewise:ListAssociatedAssets",
                "iotsitewise:DescribeAssetProperty",
                "iotsitewise:GetAssetPropertyValue",
                "iotsitewise:GetAssetPropertyValueHistory",
                "iotsitewise:GetAssetPropertyAggregates",
                "iotsitewise:BatchPutAssetPropertyValue",
                "iotsitewise:ListAssetRelationships",
                "iotsitewise:DescribeAssetModel",
                "iotsitewise:ListAssetModels",
                "iotsitewise:UpdateAssetModel",
                "iotsitewise:UpdateAssetModelPropertyRouting",
                "sso-directory:DescribeUsers",
                "sso-directory:DescribeUser",
                "iotevents:DescribeAlarmModel",
                "iotevents:ListTagsForResource"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotevents:BatchAcknowledgeAlarm",
                "iotevents:BatchSnoozeAlarm",
                "iotevents:BatchEnableAlarm",
                "iotevents:BatchDisableAlarm"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "iotevents:keyValue": "false"
                }
            }
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotevents:CreateAlarmModel",
                "iotevents:TagResource"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/iotsitewisemonitor": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotevents:UpdateAlarmModel",
                "iotevents:DeleteAlarmModel"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/iotsitewisemonitor": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "iotevents.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

For more information about the required permissions for alarms, see Setting up permissions for AWS IoT Events alarms.

When a portal user signs in, SiteWise Monitor creates a session policy based on the intersection of the service role and that user's access policies. Access policies define identities' level of access to your portals and projects. For more information about portal permissions and access policies, see Administering your SiteWise Monitor portals and CreateAccessPolicy.

Managing the SiteWise Monitor service role (console)

The AWS IoT SiteWise console facilitates the management of the SiteWise Monitor service role for portals. Upon creating a portal, the console checks for existing roles suitable for attachment. If none are available, the console can create and configure a service role for you. For more information, see Creating a portal.

Topics

- Finding a portal's service role (console)
- Creating a SiteWise Monitor service role (AWS IoT SiteWise console)
- Creating a SiteWise Monitor service role (IAM console)
- Changing a portal's service role (console)

Finding a portal's service role (console)

Use the following steps to find the service role attached to a SiteWise Monitor portal.

To find a portal's service role

1. Navigate to the AWS IoT SiteWise console.
2. In the left navigation pane, choose **Portals**.
3. Choose the portal for which you want to find the service role.

   The role attached to the portal appears under **Permissions**, **Service role**.

Creating a SiteWise Monitor service role (AWS IoT SiteWise console)

When you create a SiteWise Monitor portal, you can create a service role for your portal. For more information, see Creating a portal.

You can also create a service role for an existing portal in the AWS IoT SiteWise console. This replaces the portal's existing service role.

To create a service role for an existing portal

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Portals**.
3. Choose the portal for which you want to create a new service role.
4. Under **Portal details**, choose **Edit**.
5. Under **Permissions**, choose **Create and use a new service role** from the list.
6. Enter a name for your new role.
7. Choose **Save**.

Creating a SiteWise Monitor service role (IAM console)

You can create a service role from the service role template in the IAM console. This role template includes the AWSIoTSiteWiseMonitorPortalAccess managed policy and specifies SiteWise Monitor as a trusted entity.

To create a service role from the portal service role template

1. Navigate to the IAM console.
2. In the navigation pane, choose **Roles**.
3. Choose **Create role**.
4. In **Choose a use case**, choose **IoT SiteWise**.
5. In **Select your use case**, choose **IoT SiteWise Monitor - Portal**.
6. Choose **Next: Permissions**.
7. Choose **Next: Tags**.
8. Choose **Next: Review**.
9. Enter a **Role name** for the new service role.
10. Choose **Create role**.

Changing a portal's service role (console)

Use the following procedure to choose a different SiteWise Monitor service role for a portal.

To change a portal's service role

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Portals**.
3. Choose the portal for which you want to change the service role.
4. Under **Portal details**, choose **Edit**.
5. Under **Permissions**, choose **Use an existing role**.
6. Choose an existing role to attach to this portal.
7. Choose **Save**.

Managing the SiteWise Monitor service role (CLI)

You can use the AWS CLI for the following portal service role management tasks:

Topics

- Finding a portal's service role (CLI)
- Creating the SiteWise Monitor service role (CLI)

Finding a portal's service role (CLI)

To find the service role attached to a SiteWise Monitor portal, run the following command to list all of your portals in the current Region.

```
aws iotsitewise list-portals
```

The operation returns a response that contains your portal summaries in the following format.

```
{
  "portalSummaries": [
    {
      "id": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
      "name": "WindFarmPortal",
      "description": "A portal that contains wind farm projects for Example Corp.",
      "roleArn": "arn:aws:iam::123456789012:role/service-role/role-name",
      "startUrl": "https://a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE.app.iotsitewise.aws",
      "creationDate": "2020-02-04T23:01:52.90248068Z",
      "lastUpdateDate": "2020-02-04T23:01:52.90248078Z"
    }
```

```
  ]
}
```

You can also use the DescribePortal operation to find your portal's role if you know the ID of your portal.

Creating the SiteWise Monitor service role (CLI)

Use the following steps to create a new SiteWise Monitor service role.

To create a SiteWise Monitor service role

1. Create a role with a trust policy that allows SiteWise Monitor to assume the role. This example creates a role named **`MySiteWiseMonitorPortalRole`** from a trust policy stored in a JSON string.

   Linux, macOS, or Unix

   ```
   aws iam create-role --role-name MySiteWiseMonitorPortalRole --assume-role-
   policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "monitor.iotsitewise.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }'
   ```

   Windows command prompt

   ```
   aws iam create-role --role-name MySiteWiseMonitorPortalRole --assume-role-
   policy-document "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow
   \",\"Principal\":{\"Service\":\"monitor.iotsitewise.amazonaws.com\"},\"Action\":
   \"sts:AssumeRole\"}]}"
   ```

2. Copy the role ARN from the role metadata in the output. When you create a portal, you use this ARN to associate the role with your portal. For more information about creating a portal, see CreatePortal in the *AWS IoT SiteWise API Reference*.

3. Attach the `AWSIoTSiteWiseMonitorPortalAccess` policy to the role, or attach a policy that defines equivalent permissions.

```
aws iam attach-role-policy --role-name MySiteWiseMonitorPortalRole --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSIoTSiteWiseMonitorPortalAccess
```

To attach a service role to an existing portal

1. To retrieve the portal's existing details, run the following command. Replace *`portal-id`* with the ID of the portal.

```
aws iotsitewise describe-portal --portal-id portal-id
```

   The operation returns a response that contains the portal's details in the following format.

```
{
    "portalId": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
    "portalArn": "arn:aws:iotsitewise:region:account-id:portal/a1b2c3d4-5678-90ab-
cdef-aaaaaEXAMPLE",
    "portalName": "WindFarmPortal",
    "portalDescription": "A portal that contains wind farm projects for Example
 Corp.",
    "portalClientId": "E-1a2b3c4d5e6f_sn6tbqHVzLWVEXAMPLE",
    "portalStartUrl": "https://a1b2c3d4-5678-90ab-cdef-
aaaaaEXAMPLE.app.iotsitewise.aws",
    "portalContactEmail": "support@example.com",
    "portalStatus": {
        "state": "ACTIVE"
    },
    "portalCreationDate": "2020-04-29T23:01:52.90248068Z",
    "portalLastUpdateDate": "2020-04-29T00:28:26.103548287Z",
    "roleArn": "arn:aws:iam::123456789012:role/service-role/
AWSIoTSiteWiseMonitorServiceRole_1aEXAMPLE"
}
```

2. To attach a service role to a portal, run the following command. Replace *`role-arn`* with the service role ARN, and replace the remaining parameters with the portal's existing values.

```
aws iotsitewise update-portal \
  --portal-id portal-id \
```

```
  --role-arn role-arn \
  --portal-name portal-name \
  --portal-description portal-description \
  --portal-contact-email portal-contact-email
```

SiteWise Monitor updates to AWSIoTSiteWiseMonitorServiceRole

You can view details about updates to **AWSIoTSiteWiseMonitorServiceRole** for SiteWise Monitor, beginning from when this service began tracking the changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the AWS IoT SiteWise Document history page.

Change	Description	Date
AWSIoTSiteWiseMonitorPortalAccess – Updated policy	AWS IoT SiteWise updated the AWSIoTSiteWiseMonitorPortalAccess managed policy for the alarms feature.	May 27, 2021
AWS IoT SiteWise started tracking changes	AWS IoT SiteWise started tracking changes for its service role.	December 15, 2020

Setting up permissions for AWS IoT Events alarms

When you use an AWS IoT Events alarm model to monitor an AWS IoT SiteWise asset property, you must have the following IAM permissions:

- An AWS IoT Events service role that allows AWS IoT Events to send data to AWS IoT SiteWise. For more information, see Identity and access management for AWS IoT Events in the *AWS IoT Events Developer Guide*.
- You must have the following AWS IoT SiteWise action permissions: `iotsitewise:DescribeAssetModel` and `iotsitewise:UpdateAssetModelPropertyRouting`. These permissions allow AWS IoT SiteWise to send asset property values to AWS IoT Events alarm models.

For more information, see Resource-based policies in the *IAM User Guide*.

Required action permissions

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**. The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy.

Before you define an AWS IoT Events alarm model, you must grant the following permissions that allow AWS IoT SiteWise to send asset property values to the alarm model.

- `iotsitewise:DescribeAssetModel` – Allows AWS IoT Events to check if an asset property exists.
- `iotsitewise:UpdateAssetModelPropertyRouting` – Allows AWS IoT SiteWise to automatically create subscriptions that enable AWS IoT SiteWise to send data to AWS IoT Events.

For more information about AWS IoT SiteWise supported actions, see Actions defined by AWS IoT SiteWise in the *Service Authorization Reference*.

Example Example permissions policy 1

The following policy allows AWS IoT SiteWise to send asset property values to any AWS IoT Events alarm models.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotevents:CreateAlarmModel",
                "iotevents:UpdateAlarmModel"
            ],
            "Resource": "arn:aws:iotevents:us-east-1:123456789012:alarmModel/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:DescribeAssetModel",
                "iotsitewise:UpdateAssetModelPropertyRouting"
            ],
            "Resource": "arn:aws:iotsitewise:us-east-1:123456789012:asset-model/*"
        }
```

```
    ]
}
```

Example Example permissions policy 2

The following policy allows AWS IoT SiteWise to send values of a specified asset property to a specified AWS IoT Events alarm model.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotevents:CreateAlarmModel",
                "iotevents:UpdateAlarmModel"
            ],
            "Resource": "arn:aws:iotevents:us-east-1:123456789012:alarmModel/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:DescribeAssetModel"
            ],
            "Resource": "arn:aws:iotsitewise:us-east-1:123456789012:asset-model/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:UpdateAssetModelPropertyRouting"
            ],
            "Resource": [
                "arn:aws:iotsitewise:us-east-1:123456789012:asset-model/12345678-90ab-
cdef-1234-567890abcdef"
            ],
            "Condition": {
                "StringLike": {
                    "iotsitewise:propertyId": "abcdef12-3456-7890-abcd-ef1234567890",
                    "iotevents:alarmModelArn": "arn:aws:iotevents:us-
east-1:123456789012:alarmModel/MyAlarmModel"
                }
            }
        }
```

```
    ]
}
```

(Optional) ListInputRoutings permission

When you update or delete an asset model, AWS IoT SiteWise can check if an alarm model in AWS IoT Events is monitoring an asset property associated with this asset model. This prevents you from deleting an asset property that an AWS IoT Events alarm is currently using. To enable this feature in AWS IoT SiteWise, you must have the `iotevents:ListInputRoutings` permission. This permission allows AWS IoT SiteWise to make calls to the ListInputRoutings API operation supported by AWS IoT Events.

> **Note**
>
> We strongly recommend that you add the `ListInputRoutings` permission.

Example Example permissions policy

The following policy allows you to update and delete asset models, and use the `ListInputRoutings` API in AWS IoT SiteWise.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:UpdateAssetModel",
                "iotsitewise:DeleteAssetModel",
                "iotevents:ListInputRoutings"
            ],
            "Resource": "arn:aws:iotsitewise:us-east-1:123456789012:asset-model/*"
        }
    ]
}
```

Required permissions for SiteWise Monitor

If you want to use the alarms feature in SiteWise Monitor portals, you must update the SiteWise Monitor service role with the following policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:DescribePortal",
                "iotsitewise:CreateProject",
                "iotsitewise:DescribeProject",
                "iotsitewise:UpdateProject",
                "iotsitewise:DeleteProject",
                "iotsitewise:ListProjects",
                "iotsitewise:BatchAssociateProjectAssets",
                "iotsitewise:BatchDisassociateProjectAssets",
                "iotsitewise:ListProjectAssets",
                "iotsitewise:CreateDashboard",
                "iotsitewise:DescribeDashboard",
                "iotsitewise:UpdateDashboard",
                "iotsitewise:DeleteDashboard",
                "iotsitewise:ListDashboards",
                "iotsitewise:CreateAccessPolicy",
                "iotsitewise:DescribeAccessPolicy",
                "iotsitewise:UpdateAccessPolicy",
                "iotsitewise:DeleteAccessPolicy",
                "iotsitewise:ListAccessPolicies",
                "iotsitewise:DescribeAsset",
                "iotsitewise:ListAssets",
                "iotsitewise:ListAssociatedAssets",
                "iotsitewise:DescribeAssetProperty",
                "iotsitewise:GetAssetPropertyValue",
                "iotsitewise:GetAssetPropertyValueHistory",
                "iotsitewise:GetAssetPropertyAggregates",
                "iotsitewise:BatchPutAssetPropertyValue",
                "iotsitewise:ListAssetRelationships",
                "iotsitewise:DescribeAssetModel",
                "iotsitewise:ListAssetModels",
                "iotsitewise:UpdateAssetModel",
                "iotsitewise:UpdateAssetModelPropertyRouting",
                "sso-directory:DescribeUsers",
                "sso-directory:DescribeUser",
                "iotevents:DescribeAlarmModel",
                "iotevents:ListTagsForResource"
            ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotevents:BatchAcknowledgeAlarm",
                "iotevents:BatchSnoozeAlarm",
                "iotevents:BatchEnableAlarm",
                "iotevents:BatchDisableAlarm"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "iotevents:keyValue": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotevents:CreateAlarmModel",
                "iotevents:TagResource"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/iotsitewisemonitor": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotevents:UpdateAlarmModel",
                "iotevents:DeleteAlarmModel"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/iotsitewisemonitor": "false"
                }
            }
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "iotevents.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it shouldn't otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource policies to limit the permissions that AWS IoT SiteWise gives another service to the resource. If the `aws:SourceArn` value doesn't contain the account ID, such as an Amazon S3 bucket Amazon Resource Name (ARN), you must use both global condition context keys to limit permissions. If you use both global condition context keys and the `aws:SourceArn` value contains the account ID, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement.

- Use `aws:SourceArn` if you want only one resource to be associated with the cross-service access.
- Use `aws:SourceAccount` if you want to allow any resource in that account to be associated with the cross-service use.

The value of `aws:SourceArn` must be the AWS IoT SiteWise customer resource that is associated with the `sts:AssumeRole` request.

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the full ARN of the resource. If you don't know the full ARN of the resource or if you're specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcards (*) for the unknown portions of the ARN. For example, `arn:aws:servicename:*:123456789012:*`.

Example – Confused Deputy Prevention

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in AWS IoT SiteWise to prevent the confused deputy problem.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "iotsitewise.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Resource": [
      "arn:aws:iotsitewise:::ResourceName/*"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:iotsitewise:*:123456789012:*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

Troubleshooting AWS IoT SiteWise identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS IoT SiteWise and AWS Identity and Access Management (IAM).

Topics

I am not authorized to perform an action in AWS IoT SiteWise

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about an asset but does not have `iotsitewise:DescribeAsset` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iotsitewise:DescribeAsset on resource: a1b2c3d4-5678-90ab-cdef-22222EXAMPLE
```

In this case, Mateo asks his administrator to update his policies to allow him to access the asset resource with ID `a1b2c3d4-5678-90ab-cdef-22222EXAMPLE` using the `iotsitewise:DescribeAsset` action.

I am not authorized to perform `iam:PassRole`

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to AWS IoT SiteWise.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in AWS IoT SiteWise. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my AWS IoT SiteWise resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether AWS IoT SiteWise supports these features, see How AWS IoT SiteWise works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Compliance validation for AWS IoT SiteWise

AWS IoT SiteWise is not in scope of any AWS compliance programs.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading reports in AWS Artifact.

Your compliance responsibility when using AWS IoT SiteWise is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating resources with rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.
- Ten security golden rules for Industrial IoT solutions – This blog post introduces ten golden rules that help secure your industrial control systems (ICS), industrial Internet of Things (IIoT), and cloud environments.
- Security Best Practices for Manufacturing OT – This whitepaper describes security best practices to design, deploy, and architect these on-premises hybrid manufacturing workloads for the AWS Cloud.

Resilience in AWS IoT SiteWise

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

AWS IoT SiteWise is fully managed and uses highly available and durable AWS services, such as Amazon S3 and Amazon EC2. To ensure availability in the event of an availability zone disruption, AWS IoT SiteWise operates across multiple availability zones.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, AWS IoT SiteWise offers several features to help support your data resiliency and backup needs:

- You can publish property value updates to AWS IoT Core through MQTT messages, then configure rules to act upon that data. With this feature, you can back up data in other AWS services such as Amazon S3 and Amazon DynamoDB. For more information, see Interacting with other AWS services and Export data to Amazon S3 with asset property notifications.
- You can use the AWS IoT SiteWise `Get*` APIs to retrieve and backup historical asset property data. For more information, see Querying historical asset property values.
- You can use the AWS IoT SiteWise `Describe*` APIs to retrieve the definitions for your resources, such as assets and models. You can backup these definitions and later use them to recreate your resources. For more information, see the *AWS IoT SiteWise API Reference*.

Infrastructure security in AWS IoT SiteWise

As a managed service, AWS IoT SiteWise is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access AWS IoT SiteWise through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

SiteWise Edge gateways, which run on AWS IoT Greengrass, use X.509 certificates and cryptographic keys to connect and authenticate to the AWS Cloud. For more information, see Device authentication and authorization for AWS IoT Greengrass in the *AWS IoT Greengrass Version 1 Developer Guide*.

Configuration and vulnerability analysis

IoT fleets can consist of large numbers of devices that have diverse capabilities, are long-lived, and are geographically distributed. These characteristics make fleet setup complex and error-prone. Because devices usually have limited processing power, memory, and storage, they can't always support encryption and other security measures. Also, devices often use software with known vulnerabilities. These factors make IoT fleets an attractive target for hackers and make it difficult to secure your device fleet on an ongoing basis.

AWS IoT Device Defender addresses these challenges by providing tools to identify security issues and deviations from best practices. Use AWS IoT Device Defender to analyze, audit, and monitor connected devices to detect abnormal behavior, and mitigate security risks. AWS IoT Device Defender can audit device fleets to ensure they adhere to security best practices and detect abnormal behavior on devices. This makes it possible to enforce consistent security policies across your AWS IoT device fleet and respond quickly when devices are compromised. For more information, see AWS IoT Device Defender in the *AWS IoT Developer Guide*.

If you use SiteWise Edge gateways to ingest data to the service, it's your responsibility to configure and maintain your SiteWise Edge gateway's environment. This responsibility includes upgrading to the latest versions of the SiteWise Edge gateway's system software, AWS IoT Greengrass software, and the AWS IoT SiteWise connector. For more information, see Configure the AWS IoT Greengrass core in the *AWS IoT Greengrass Version 1 Developer Guide* and Upgrading a connector.

VPC endpoints

An *interface VPC endpoint* establishes a private connection between your virtual private cloud (VPC) and AWS IoT SiteWise. AWS PrivateLink powers interface endpoints, enabling private access to AWS IoT SiteWise API operations. You can bypass the need for an internet gateway, NAT device, VPN connection, or AWS Direct Connect. Instances in your VPC don't need public IP addresses to communicate with AWS IoT SiteWise API operations. Traffic between your VPC and AWS IoT SiteWise doesn't leave the AWS network.

Each interface endpoint is represented by one or more elastic network interfaces in your subnets.

Before you set up an interface VPC endpoint for AWS IoT SiteWise, review the Interface endpoint properties and limitations in the *Amazon VPC User Guide*.

For more information, see Interface VPC endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*.

Supported API operations for VPC endpoints

AWS IoT SiteWise supports making calls to the following AWS IoT SiteWise API operations from your VPC:

- For all the **data plane** API operations, use the following endpoint: Replace *region* with your AWS Region

  ```
  data.iotsitewise.region.amazonaws.com
  ```

  The data plane API operations include the following:

  - BatchGetAssetPropertyValue
  - BatchGetAssetPropertyValueHistory
  - BatchPutAssetPropertyValue
  - GetAssetPropertyAggregates
  - GetAssetPropertyValue
  - GetAssetPropertyValueHistory
  - GetInterpolatedAssetPropertyValues
- For the **control plane** API operations that you use to manage asset models, assets, SiteWise Edge gateways, tags, and account configurations, use the following endpoint. Replace *region* with your AWS Region.

  ```
  api.iotsitewise.region.amazonaws.com
  ```

  The supported control plane API operations include the following:

  - AssociateAssets
  - CreateAsset
  - CreateAssetModel
  - DeleteAsset
  - DeleteAssetModel
  - DeleteDashboard
  - DescribeAsset
  - DescribeAssetModel
  - DescribeAssetProperty

- DescribeDashboard
- DescribeLoggingOptions
- DisassociateAssets
- ListAssetModels
- ListAssetRelationships
- ListAssets
- ListAssociatedAssets
- PutLoggingOptions
- UpdateAsset
- UpdateAssetModel
- UpdateAssetProperty
- CreateGateway
- DeleteGateway
- DescribeDefaultEncryptionConfiguration
- DescribeGateway
- DescribeGatewayCapabilityConfiguration
- DescribeStorageConfiguration
- ListGateways
- ListTagsForResource
- UpdateGateway
- UpdateGatewayCapabilityConfiguration
- PutDefaultEncryptionConfiguration
- PutStorageConfiguration
- TagResource
- UntagResource

Note

The interface VPC endpoint for the **control plane** API operations currently doesn't support making calls to the following SiteWise Monitor API operations:

- BatchAssociateProjectAssets

- BatchDisassociateProjectAssets
- CreateAccessPolicy
- CreateDashboard
- CreatePortal
- CreateProject
- DeleteAccessPolicy
- DeletePortal
- DeleteProject
- DescribeAccessPolicy
- DescribePortal
- DescribeProject
- ListAccessPolicies
- ListDashboards
- ListPortals
- ListProjects
- ListProjectAssets
- UpdateAccessPolicy
- UpdateDashboard
- UpdatePortal
- UpdateProject

Creating an interface VPC endpoint for AWS IoT SiteWise

To create a VPC endpoint for the AWS IoT SiteWise service, use either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Creating an interface endpoint in the *Amazon VPC User Guide*.

Create a VPC endpoint for AWS IoT SiteWise by using one of the following service names:

- For the **data plane** API operations, use the following service name:

```
com.amazonaws.region.iotsitewise.data
```

- For the **control plane** API operations, use the following service name:

```
com.amazonaws.region.iotsitewise.api
```

Accessing AWS IoT SiteWise through an interface VPC endpoint

When you create an interface endpoint, we generate endpoint-specific DNS hostnames that you can use to communicate with AWS IoT SiteWise. The private DNS option is enabled by default. For more information, see Using private hosted zones in the *Amazon VPC User Guide*.

If you enable private DNS for the endpoint, you can make API requests to AWS IoT SiteWise through one of the following VPC endpoints.

- For the **data plane** API operations, use the following endpoint: Replace *region* with your AWS Region.

```
data.iotsitewise.region.amazonaws.com
```

- For the **control plane** API operations, use the following endpoint: Replace *region* with your AWS Region.

```
api.iotsitewise.region.amazonaws.com
```

If you disable private DNS for the endpoint, you must do the following to access AWS IoT SiteWise through the endpoint:

1. Specify the VPC endpoint url in API requests.
   - For the **data plane** API operations, use the following endpoint url. Replace *vpc-endpoint-id* and *region* with your VPC endpoint ID and Region.

```
vpc-endpoint-id.data.iotsitewise.region.vpce.amazonaws.com
```

   - For the **control plane** API operations, use the following endpoint url. Replace *vpc-endpoint-id* and *region* with your VPC endpoint ID and Region.

```
vpc-endpoint-id.api.iotsitewise.region.vpce.amazonaws.com
```

2. Disable host prefix injection. The AWS CLI and AWS SDKs prepend the service endpoint with various host prefixes when you call each API operation. This feature causes the AWS CLI and AWS SDKs to produce URLs that are not valid for AWS IoT SiteWise when you specify a VPC endpoint.

   > **Important**
   >
   > You can't disable host prefix injection in the AWS CLI or the AWS Tools for PowerShell. This means that if you disable private DNS, then you can't use these tools to access AWS IoT SiteWise through the VPC endpoint. Enable private DNS to use the AWS CLI or the AWS Tools for PowerShell to access AWS IoT SiteWise through the endpoint.

   For more information about how to disable host prefix injection in the AWS SDKs, see the following documentation sections for each SDK:

   - AWS SDK for C++
   - AWS SDK for Go
   - AWS SDK for Go v2
   - AWS SDK for Java
   - AWS SDK for Java 2.x
   - AWS SDK for JavaScript
   - AWS SDK for .NET
   - AWS SDK for PHP
   - AWS SDK for Python (Boto3)
   - AWS SDK for Ruby

For more information, see Accessing a service through an interface endpoint in the *Amazon VPC User Guide*.

Creating a VPC endpoint policy for AWS IoT SiteWise

You can attach an endpoint policy to your VPC endpoint that controls access to AWS IoT SiteWise. The policy specifies the following information:

- The principal that can perform operations.

- The operations that can be performed.
- The resources on which operations can be performed.

For more information, see Controlling access to services with VPC endpoints in the *Amazon VPC User Guide*.

Example: VPC endpoint policy for AWS IoT SiteWise actions

The following is an example of an endpoint policy for AWS IoT SiteWise. When attached to an endpoint, this policy grants access to the listed AWS IoT SiteWise actions for the user *iotsitewiseadmin* in AWS account *123456789012* on the specified asset.

```
{
    "Statement": [
        {
            "Action": [
                "iotsitewise:CreateAsset",
                "iotsitewise:ListGateways",
                "iotsitewise:ListTagsForResource"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iotsitewise:us-west-2:123456789012:asset/
a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
            "Principal": {
                "AWS": [
                    "123456789012:user/iotsitewiseadmin"
                ]
            }
        }
    ]
}
```

Security best practices for AWS IoT SiteWise

This topic contains security best practices for AWS IoT SiteWise.

Use authentication credentials on your OPC-UA servers

Require authentication credentials to connect to your OPC-UA servers. Consult the documentation for your servers to do so. Then, to allow your SiteWise Edge gateway to connect to your OPC-UA

servers, add server authentication secrets to your SiteWise Edge gateway. For more information, see Configuring source authentication.

Use encrypted communication modes for your OPC-UA servers

Choose a non-deprecated, encrypted message security mode when you configure your OPC-UA sources for your SiteWise Edge gateway. This helps secure your industrial data as it moves from your OPC-UA servers to the SiteWise Edge gateway. For more information, see Data in transit over the local network and Configuring data sources.

Keep your components up to date

If you use SiteWise Edge gateways to ingest data to the service, it's your responsibility to configure and maintain your SiteWise Edge gateway's environment. This responsibility includes upgrading to the latest versions of the gateway's system software, AWS IoT Greengrass software, and connectors.

> **Note**
>
> The AWS IoT SiteWise Edge connector stores secrets on your file system. These secrets control who can view the data cached within your SiteWise Edge gateway. It's strongly recommended that you turn on disk or file-system encryption for the system running your SiteWise Edge gateway.

Encrypt your SiteWise Edge gateway's file system

Encrypt and secure your SiteWise Edge gateway, so your industrial data is secure as it moves through the SiteWise Edge gateway. If your SiteWise Edge gateway has a hardware security module, you can configure AWS IoT Greengrass to secure your SiteWise Edge gateway. For more information, see Hardware security integration in the *AWS IoT Greengrass Version 1 Developer Guide*. Otherwise, consult the documentation for your operating system to learn how to encrypt and secure your file system.

Secure access to your edge configuration

Don't share your edge console application password or your SiteWise Monitor application password. Don't put this password in places where anyone can see them. Implement a healthy password rotation policy by configuring an appropriate expiration for your password.

Grant SiteWise Monitor users minimum possible permissions

Follow the principle of least privilege by using the minimum set of access policy permissions for your portal users.

- When you create a portal, define a role that allows the minimum set of assets needed for that portal. For more information, see Using service roles for AWS IoT SiteWise Monitor.
- When you and your portal administrators create and share projects, use the minimum set of assets needed for that project.
- When an identity no longer needs access to a portal or project, remove them from that resource. If that identity is no longer applicable to your organization, delete that identity from your identity store.

The least principle best practice also applies to IAM roles. For more information, see Policy best practices.

Don't expose sensitive information

You should prevent the logging of credentials and other sensitive information, such as personally identifiable information (PII). We recommend that you implement the following safeguards even though access to local logs on a SiteWise Edge gateway requires root privileges and access to CloudWatch Logs requires IAM permissions.

- Don't use sensitive information in names, descriptions, or properties of your assets or models.
- Don't use sensitive information in SiteWise Edge gateway or source names.
- Don't use sensitive information in names or descriptions of your portals, projects, or dashboards.

Follow AWS IoT Greengrass security best practices

Follow AWS IoT Greengrass security best practices for your SiteWise Edge gateway. For more information, see Security best practices in the *AWS IoT Greengrass Version 1 Developer Guide*.

See also

- Security best practices in the *AWS IoT Developer Guide*
- Ten security golden rules for Industrial IoT solutions

Logging and monitoring in AWS IoT SiteWise

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS IoT SiteWise and your other AWS solutions. AWS IoT SiteWise supports the following monitoring tools to watch the service, report when something is wrong, and take automatic actions when appropriate:

- *Amazon CloudWatch* monitors your AWS resources and the applications that you run on AWS in real time. Collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a certain threshold. For example, you can have CloudWatch track CPU usage or other metrics of your Amazon EC2 instances and automatically launch new instances when needed. For more information, see the Amazon CloudWatch User Guide.
- *Amazon CloudWatch Logs* monitors, stores, and accesses your log files from SiteWise Edge gateways, CloudTrail, and other sources. CloudWatch Logs can monitor information in the log files and notify you when certain thresholds are met. You can also archive your log data in highly durable storage. For more information, see the Amazon CloudWatch Logs User Guide.
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account. Then CloudTrail delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.

Topics

Monitoring with Amazon CloudWatch Logs

Configure AWS IoT SiteWise to log information to CloudWatch Logs to monitor and troubleshoot the service.

When you use the AWS IoT SiteWise console, AWS IoT SiteWise creates a service-linked role that allows the service to log information on your behalf. If you don't use the AWS IoT SiteWise console,

you must create a service-linked role manually to receive logs. For more information, see Creating a service-linked role for AWS IoT SiteWise.

You must have a resource policy that allows AWS IoT SiteWise to put log events into CloudWatch streams. To create and update a resource policy for CloudWatch Logs, run the following command. Replace *logging-policy-name* with the name of the policy to create.

```
aws logs put-resource-policy --policy-name logging-policy-name --policy-
document "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Sid\":
 \"IoTSiteWiseToCloudWatchLogs\", \"Effect\": \"Allow\", \"Principal\": { \"Service\":
 [ \"iotsitewise.amazonaws.com\" ] }, \"Action\":\"logs:PutLogEvents\", \"Resource\":
 \"*\" } ] }"
```

CloudWatch Logs also supports aws:SourceArn and aws:SourceAccount condition context keys. These condition context keys are optional.

To create or update a resource policy that allows AWS IoT SiteWise to only put logs associated with the specified AWS IoT SiteWise resource into CloudWatch streams, run the command and do the following:

- Replace *logging-policy-name* with the name of the policy to create.
- Replace *source-ARN* with the ARN of your AWS IoT SiteWise resource, such as an asset model or asset. To find the ARN for each AWS IoT SiteWise resource type, see Resource types defined by AWS IoT SiteWise in the *Service Authorization Reference*.
- Replace *account-ID* with the AWS account ID associated with the specified AWS IoT SiteWise resource.

```
aws logs put-resource-policy --policy-name logging-policy-name --policy-
document "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Sid\":
 \"IoTSiteWiseToCloudWatchLogs\", \"Effect\": \"Allow\", \"Principal\": { \"Service
\": [ \"iotsitewise.amazonaws.com\" ] }, \"Action\":\"logs:PutLogEvents\", \"Resource
\": \"*\", \"Condition\":{\"StringLike\":{\"aws:SourceArn\":[\"source-ARN\"],
\"aws:SourceAccount\":[\"account-ID\"]}}}]}"
```

By default, AWS IoT SiteWise doesn't log information to CloudWatch Logs. To activate logging, choose a logging level other than **Disabled** (OFF). AWS IoT SiteWise supports the following logging levels:

- OFF – Logging is turned off.

- `ERROR` – Errors are logged.
- `INFO` – Errors and informational messages are logged.

You can configure SiteWise Edge gateways to log information to CloudWatch Logs through AWS IoT Greengrass. For more information, see Monitoring SiteWise Edge gateway logs.

You can also configure AWS IoT Core to log information to CloudWatch Logs if you are troubleshooting an AWS IoT SiteWise rule action. For more information, see Troubleshooting an AWS IoT SiteWise rule action.

Contents

Managing logging in AWS IoT SiteWise

Use the AWS IoT SiteWise console or AWS CLI for the following logging configuration tasks.

Finding your logging level

Console

Use the following procedure to find your current logging level in the AWS IoT SiteWise console.

To find your current AWS IoT SiteWise logging level

1. Navigate to the AWS IoT SiteWise console.
2. In the left navigation pane, choose **Logging options**.

   The current logging status appears under **Logging status**. If logging is activated, the current logging level appears under **Level of verbosity**.

AWS CLI

Run the following command to find your current AWS IoT SiteWise logging level with the AWS CLI.

```
aws iotsitewise describe-logging-options
```

The operation returns a response that contains your logging level in the following format.

```
{
  "loggingOptions": {
    "level": "String"
  }
}
```

Changing your logging level

Use the following procedure to change your logging level in the AWS IoT SiteWise console or using AWS CLI.

Console

To change your AWS IoT SiteWise logging level

1. Navigate to the AWS IoT SiteWise console.
2. In the left navigation pane, choose **Logging options**.
3. Choose **Edit**.
4. Choose the **Level of verbosity** to activate.
5. Choose **Save**.

AWS CLI

Run the following AWS CLI command to change your AWS IoT SiteWise logging level. Replace *logging-level* with the logging level you want.

```
aws iotsitewise put-logging-options --logging-options level=logging-level
```

Example: AWS IoT SiteWise log file entries

Each AWS IoT SiteWise log entry includes event information and relevant resources for that event, so you can understand and analyze log data.

The following example shows a CloudWatch Logs entry that AWS IoT SiteWise logs when you successfully create an asset model.

```
{
  "eventTime": "2020-05-05T00:10:22.902Z",
  "logLevel": "INFO",
  "eventType": "AssetModelCreationSuccess",
  "message": "Successfully created asset model.",
  "resources": {
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
  }
}
```

Monitoring SiteWise Edge gateway logs

You can configure your AWS IoT SiteWise Edge gateway to log information to Amazon CloudWatch Logs or the local file system.

Topics

- Using Amazon CloudWatch Logs
- Using service logs
- Using event logs

Using Amazon CloudWatch Logs

You can configure your SiteWise Edge gateway to send logs to CloudWatch Logs. For more information, see Enable logging for CloudWatch Logs in the *AWS IoT Greengrass Version 2 Developer Guide*.

To configure and access CloudWatch Logs (Console)

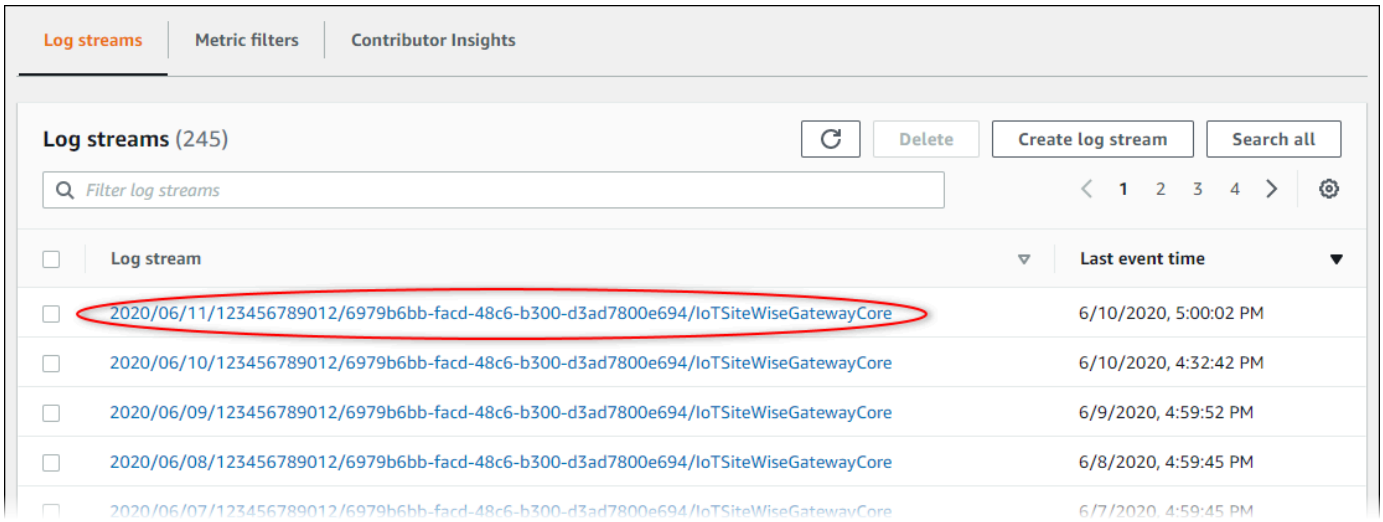
1. Navigate to the CloudWatch console.
2. In the navigation pane, choose **Log groups**.
3. You can find the AWS IoT SiteWise component logs in the following log groups:

   - `/aws/greengrass/UserComponent/`*`region`*`/aws.iot.SiteWiseEdgeCollectorOpcua` – The logs for the SiteWise Edge gateway's component that collects data from the SiteWise Edge gateway's OPC-UA sources.

- `/aws/greengrass/UserComponent/region/aws.iot.SiteWiseEdgePublisher` – The logs for the SiteWise Edge gateway's component that publishes OPC-UA data streams to AWS IoT SiteWise.

Choose the log group for the function to debug.

4. Choose a log stream that has a name that ends with the name of your AWS IoT Greengrass group. By default, CloudWatch displays the most recent log stream first.



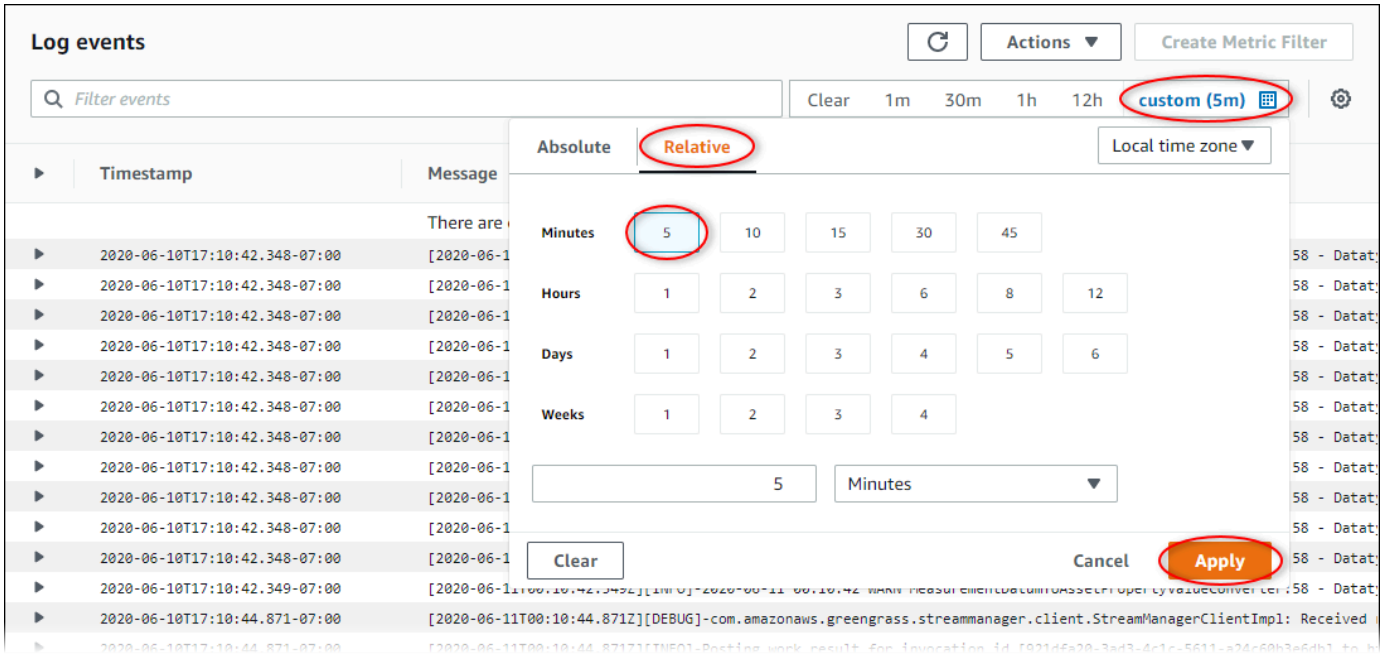
5. To show logs from the last 5 minutes, do the following:

   a. Choose **custom** in the upper-right corner.

   b. Choose **Relative**.

   c. Choose **5** minutes.

   d. Choose **Apply**.



6. (Optional) To see fewer logs, you can choose **1m** from the upper-right corner.
7. Scroll to the bottom of the log entries to show the most recent logs.

Using service logs

SiteWise Edge gateway devices include service log files to help debug issues. The following sections will help you find and utilize the service log files for the AWS IoT SiteWise OPC-UA Collector and AWS IoT SiteWise Publisher components.

AWS IoT SiteWise OPC-UA Collector service log file

The AWS IoT SiteWise OPC-UA Collector component uses the following log file.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgeCollectorOpcua.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeCollectorOpcua.log
```

To view this component's logs

- Run the following command on the core device to view this component's log file in real time. Replace */greengrass/v2* or *C:\greengrass\v2* with the path to the AWS IoT Greengrass root folder.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.iot.SiteWiseEdgeCollectorOpcua.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeCollectorOpcua.log -Tail
   10 -Wait
  ```

AWS IoT SiteWise Publisher service log file

The AWS IoT SiteWise Publisher component uses the following log file.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgePublisher.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgePublisher.log
```

To view this component's logs

- Run the following command on the core device to view this component's log file in real time. Replace */greengrass/v2* or *C:\greengrass\v2* with the path to the AWS IoT Greengrass root folder.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.iot.SiteWiseEdgePublisher.log
  ```

Windows (PowerShell)

```
Get-Content C:\greengrass\v2\logs\aws.iot.SiteWiseEdgePublisher.log -Tail 10 -
Wait
```

Using event logs

SiteWise Edge gateway devices include events log files to help debug issues. The following sections will help you find and utilize the events log files for the AWS IoT SiteWise OPC-UA Collector and AWS IoT SiteWise Publisher components.

AWS IoT SiteWise OPC-UA Collector event logs

The AWS IoT SiteWise OPC-UA Collector component includes an events log to help customers identify and fix problems. The log file is separate from the local log file, and is found in the following location. Replace */greengrass/v2* or *C:\greengrass\v2* with the path to the AWS IoT Greengrass root folder.

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua/logs/
IotSiteWiseOpcUaCollectorEvents.log
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgeCollectorOpcua\logs
\IotSiteWiseOpcUaCollectorEvents.log
```

This log includes detailed information and troubleshooting instructions. Troubleshooting information is provided alongside the diagnostics, with a description of how to remedy the issue, and sometimes with links to further information. Diagnostic information includes the following:

- Severity level
- Timestamp
- Additional event-specific information

Example Example log

```
dataSourceConnectionSuccess:
  Summary: Successfully connected to OpcUa server
  Level: INFO
  Timestamp: '2023-06-15T21:04:16.303Z'
  Description: Successfully connected to the data source.
  AssociatedMetrics:
  - Name: FetchedDataStreams
    Description: The number of fetched data streams for this data source
    Value: 1.0
    Namespace: IoTSiteWise
    Dimensions:
    - Name: SourceName
      Value: SourceName{value=OPC-UA Server}
    - Name: ThingName
      Value: test-core
  AssociatedData:
  - Name: DataSourceTrace
    Description: Name of the data source
    Data:
    - OPC-UA Server
  - Name: EndpointUri
    Description: The endpoint to which the connection was attempted.
    Data:
    - '"opc.tcp://10.0.0.1:1234"'
```

AWS IoT SiteWise Publisher event logs

The AWS IoT SiteWise Publisher component includes an events log to help customers identify and fix problems. The log file is separate from the local log file, and is found in the following location. Replace */greengrass/v2* or *C:\greengrass\v2* with the path to the AWS IoT Greengrass root folder.

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/logs/
IotSiteWisePublisherEvents.log
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgePublisher\logs
\IotSiteWisePublisherEvents.log
```

This log includes detailed information and troubleshooting instructions. Troubleshooting information is provided alongside the diagnostics, with a description of how to remedy the issue, and sometimes with links to further information. Diagnostic information includes the following:

- Severity level
- Timestamp
- Additional event-specific information

Example Example log

```
accountBeingThrottled:
  Summary: Data upload speed slowed due to quota limits
  Level: WARN
  Timestamp: '2023-06-09T21:30:24.654Z'
  Description: The IoT SiteWise Publisher is limited to the "Rate of data points
 ingested"
    quota for a customers account. See the associated documentation and associated
    metric for the number of requests that were limited for more information. Note
    that this may be temporary and not require any change, although if the issue
 continues
    you may need to request an increase for the mentioned quota.
  FurtherInformation:
  - https://docs.aws.amazon.com/iot-sitewise/latest/userguide/quotas.html
  - https://docs.aws.amazon.com/iot-sitewise/latest/userguide/troubleshooting-
gateway.html#gateway-issue-data-streams
  AssociatedMetrics:
  - Name: TotalErrorCount
    Description: The total number of errors of this type that occurred.
    Value: 327724.0
  AssociatedData:
  - Name: AggregatePropertyAliases
    Description: The aggregated property aliases of the throttled data.
    FileLocation: /greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/./logs/data/
AggregatePropertyAliases_1686346224654.log
```

Monitoring AWS IoT SiteWise with Amazon CloudWatch metrics

You can monitor AWS IoT SiteWise using CloudWatch, which collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the Amazon CloudWatch User Guide.

AWS IoT SiteWise publishes the metrics and dimensions listed in the sections below to the `AWS/IoTSiteWise` namespace.

> **Tip**
>
> AWS IoT SiteWise publishes metrics on a one minute interval. When you view these metrics in graphs in the CloudWatch console, we recommend that you choose a **Period** of **1 minute**. This lets you see the highest available resolution of your metric data.

Topics

AWS IoT Greengrass Version 2 gateway metrics

AWS IoT SiteWise publishes the following SiteWise Edge gateway metrics. All SiteWise Edge gateway metrics are published on a one minute interval.

SiteWise Edge gateway metrics

Metric	Description
`Gateway.CpuUsage`	The CPU usage of a SiteWise Edge gateway. Unit: Percentage Dimension: None

Metric	Description
`Gateway.TotalDiskSpace`	The total disk space of a SiteWise Edge gateway. Unit: Bytes Dimension: None
`Gateway.UsedDiskSpace`	The used disk space of a SiteWise Edge gateway. Unit: Bytes Dimension: None
`Gateway.AvailableDiskSpace`	The available disk space of a SiteWise Edge gateway. Unit: Bytes Dimension: None
`Gateway.UsedPercentageDiskSpace`	The used percentage of disk space of a SiteWise Edge gateway. Unit: Bytes Dimension: None
`Gateway.TotalMemory`	The total memory of a SiteWise Edge gateway. Unit: Bytes Dimension: None
`Gateway.UsedMemory`	The used memory of a SiteWise Edge gateway. Unit: Bytes Dimension: None

Metric	Description
`Gateway.AvailableMemory`	The available memory of a SiteWise Edge gateway. Unit: Bytes Dimension: None
`Gateway.UsedPercentageMemory`	The used percentage memory of a SiteWise Edge gateway. Unit: Bytes Dimension: None
`Gateway.CloudConnectivity`	The cloud connectivity status of a SiteWise Edge gateway. Unit: None Dimension: GatewayId
`Gateway.SWE.Component.RunningStatus`	The running status of components on a SiteWise Edge gateway. Unit: None Dimension: GatewayId

OPC-UA collector metrics

Metric	Description
`OpcUaCollector.Heartbeat`	Generated every minute for each OPC-UA source (`sourceName`) connected to a SiteWise Edge gateway (`gatewayId`).

Metric	Description
	Unit: Count (1 representing the source is connected and 0 representing the source is disconnected.) Dimensions: GatewayId, SourceName
`OpcUaCollector.ActiveDataStreamCount`	The number of data streams that a SiteWise Edge gateway (`gatewayId`) subscribed to for an OPC-UA source (`sourceName`). Unit: Count Dimensions: GatewayId, SourceName, PropertyGroup
`OpcUaCollector.IncomingValuesCount`	The number of data points that a SiteWise Edge gateway (`gatewayId`) received for an OPC-UA source (`sourceName`), generated every minute. Unit: Count Dimensions: GatewayId, SourceName, PropertyGroup
`OpcUaCollector.IncomingValuesError`	The number of data points that an SiteWise Edge gateway (`gatewayId`) receives from an OPC-UA source (`sourceName`) that are not valid values. These data points are not ingested by the OpcUa Collector, generated every minute. Unit: Count Dimensions: GatewayId, SourceName, PropertyGroup

Metric	Description
`OpcUaCollector.ConversionErrors`	The number of data points that a SiteWise Edge gateway (`gatewayId`) received for an OPC-UA source (`sourceName`) which resulted in conversion errors while sending the data to AWS IoT SiteWise. These data points will not be ingested by OpcUa Collector. Unit: Count Dimensions: GatewayId, SourceName

AWS IoT SiteWise publisher metrics

Metric	Description
`IoTSiteWisePublisher.Heartbeat`	Generated every minute by the Publisher in the SiteWise Edge gateway. Unit: 1 (1 representing the Publisher is running and missing the data point representing the Publisher is not running.) Dimensions: GatewayId
`IoTSiteWisePublisher.PublishSuccessCount`	The number of data points that a SiteWise Edge gateway (`GatewayId`) successfully published to the cloud, generated every minute. Unit: Count Dimensions: GatewayId
`IoTSiteWisePublisher.PublishFailureCount`	The number of data points that a SiteWise Edge gateway (`GatewayId`) failed to publish, generated every minute.

Metric	Description
	Unit: Count Dimensions: GatewayId
`IoTSiteWisePublisher.PublishRejectedCount`	The number of data points that a SiteWise Edge gateway (`GatewayId`) rejected from the cloud side, generated every minute. Unit: Count Dimensions: GatewayId
`IoTSiteWisePublisher.DroppedCount`	The number of data points that are dropped by a SiteWise Edge gateway (`GatewayId`) and not published to the cloud, generated every minute. Unit: Count Dimensions: GatewayId

AWS IoT Greengrass Version 1 gateway metrics

AWS IoT SiteWise publishes the following SiteWise Edge gateway metrics. All SiteWise Edge gateway metrics are published on a one minute interval.

> **Important**
>
> To receive SiteWise Edge gateway metrics, you must use at least version 6 of the AWS IoT SiteWise connector on your SiteWise Edge gateway. For more information, see AWS IoT SiteWise OPC-UA collector in the *AWS IoT Greengrass Version 1 Developer Guide*.

SiteWise Edge gateway metrics

Metric	Description
`Gateway.Heartbeat`	Generated every minute for each SiteWise Edge gateway (`gatewayId`) connected. Unit: 1 (1 representing the SiteWise Edge gateway is up and missing the datapoint representing the SiteWise Edge gateway is disconnected from the cloud.) Dimension: GatewayId
`Gateway.PublishSuccessCount`	The number of data points that a SiteWise Edge gateway (`gatewayId`) successfully published. Unit: Count Dimension: GatewayId
`Gateway.PublishFailureCount`	The number of data points that a SiteWise Edge gateway (`gatewayId`) failed to publish. This metric counts errors that result from the SiteWise Edge gateway's calls to the BatchPutAssetPropertyValue operation. For more information about troubleshooting SiteWise Edge gateways, see Troubleshooting an SiteWise Edge gateway. Unit: Count Dimension: GatewayId
`Gateway.ProcessFailureCount`	The number of data points that a SiteWise Edge gateway (`gatewayId`) failed to process. This metric count errors that occur between the SiteWise Edge gateway and the SiteWise

Metric	Description
	Edge gateway's sources, including errors reported by sources. For more informati on about troubleshooting SiteWise Edge gateways, see Troubleshooting an SiteWise Edge gateway. Unit: Count Dimension: GatewayId
`Gateway.PublishRejectedCount`	The number of data points from a SiteWise Edge gateway (`gatewayId`) that are rejected. Unit: Count Dimension: GatewayId

OPC-UA related metrics

Metric	Description
`OPCUACollector.Heartbeat`	Generated every minute for each OPC-UA source (`sourceName`) connected to a SiteWise Edge gateway (`gatewayId`). Unit: Count (1 representing the source is connected and 0 representing the source is disconnected.) Dimensions: GatewayId, SourceName
`OPCUACollector.ActiveDataSt reamCount`	The number of data streams that a SiteWise Edge gateway (`gatewayId`) subscribed to for an OPC-UA source (`sourceName`). Unit: Count

Metric	Description
	Dimensions: GatewayId, SourceName, PropertyGroup
`OpcUaCollector.IncomingValuesCount`	The number of data points that a SiteWise Edge gateway (`gatewayId`) received for an OPC-UA source (`sourceName`), generated every minute. Unit: Count Dimensions: GatewayId, SourceName, PropertyGroup
`OpcUaCollector.IncomingValuesError`	The number of data points that a SiteWise Edge gateway (`gatewayId`) received from an OPC-UA source (`sourceName`) that are not valid values. These data points will not be ingested by the OpcUa Collector, generated every minute. Unit: Count Dimensions: GatewayId, SourceName, PropertyGroup
`OpcUaCollector.ConversionErrors`	The number of data points that a SiteWise Edge gateway (`gatewayId`) received for an OPC-UA source (`sourceName`) which resulted in conversion errors while sending the data to AWS IoT SiteWise. These data points will not be ingested by OpcUa Collector. Unit: Count Dimensions: GatewayId, SourceName

EIP related metrics

Metric	Description
`EIPCollector.Heartbeat`	Generated every minute for each EIP Source (`sourceName`) connected to a SiteWise Edge gateway (`gatewayId`). Unit: 1 (1 representing the source is connected and missing the datapoint representing the source is disconnected.) Dimensions: GatewayId, SourceName
`EIPCollector.IncomingValuesCount`	The number of data streams that a SiteWise Edge gateway (`gatewayId`) is subscribed to for an EIP source (`sourceName`). Unit: Count Dimensions: GatewayId, SourceName
`EIPCollector.ActiveDataStreamCount`	The number of data points that a SiteWise Edge gateway (`gatewayId`) received for an EIP source (`sourceName`). Unit: Count Dimensions: GatewayId, SourceName

Modbus related metrics

Metric	Description
`ModbusTCPCollector.Heartbeat`	Generated every minute for each Modbus Source (`sourceName`) connected to a SiteWise Edge gateway (`gatewayId`).

Metric	Description
	Unit: 1 (1 representing the Modbus source is connected and missing the datapoint representing the source is disconnected.) Dimensions: GatewayId, SourceName
`ModbusTCPCollector.IncomingValuesCount`	The number of data streams that a SiteWise Edge gateway (`gatewayId`) is subscribed to for a Modbus source (`sourceName`). Unit: Count Dimensions: GatewayId, SourceName
`ModbusTCPCollector.ActiveDataStreamCount`	The number of data points that a SiteWise Edge gateway (`gatewayId`) received for a Modbus source (`sourceName`). Unit: Count Dimensions: GatewayId, SourceName

Logging AWS IoT SiteWise API calls with AWS CloudTrail

AWS IoT SiteWise is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS IoT SiteWise. CloudTrail captures API calls for AWS IoT SiteWise as events. The calls captured include calls from the AWS IoT SiteWise console and code calls to the AWS IoT SiteWise API operations. If you create a trail, you can activate continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS IoT SiteWise. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to AWS IoT SiteWise, the IP address from which the request was made, who made the request, when it was made, and additional details.

For more information about CloudTrail, see the AWS CloudTrail User Guide.

AWS IoT SiteWise information in CloudTrail

CloudTrail is activated on your AWS account when you create the account. When supported event activity occurs in AWS IoT SiteWise, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing events with CloudTrail event history.

For an ongoing record of events in your AWS account, including events for AWS IoT SiteWise, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple Regions and Receiving CloudTrail log files from multiple accounts

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

AWS IoT SiteWise data events in CloudTrail

Data events provide information about the resource operations performed on or in a resource (for example, reading or writing to an Amazon S3 object). These are also known as data plane operations. Data events are often high-volume activities. By default, CloudTrail doesn't log data events. The CloudTrail **Event history** doesn't record data events.

Additional charges apply for data events. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

You can log data events for the AWS IoT SiteWise resource types by using the CloudTrail console, AWS CLI, or CloudTrail API operations. The table in this section shows the resource types available for AWS IoT SiteWise.

- To log data events using the CloudTrail console, create a trail or event data store to log data events, or update an existing trail or event data store to log data events.
  1. Choose **Data events** to log data events.
  2. From the **Data event type** list, choose the resource type for which you want to log data events.
  3. Choose the log selector template you want to use. You can log all data events for the resource type, log all `readOnly` events, log all `writeOnly` events, or create a custom log selector template to filter on the `readOnly`, `eventName`, and `resources.ARN` fields.
- To log data events using the AWS CLI, configure the `--advanced-event-selectors` parameter to set the `eventCategory` field equal to `Data` and the `resources.type` field equal to the resource type value (see table). You can add conditions to filter on the values of the `readOnly`, `eventName`, and `resources.ARN` fields.
  - To configure a trail to log data events, run the AWS CloudTrail put-event-selectors command. For more information, see Logging data events for trails with the AWS CLI.
  - To configure an event data store to log data events, run the AWS CloudTrail create-event-data-store command to create a new event data store to log data events, or run the AWS CloudTrail update-event-data-store command to update an existing event data store. For more information, see Logging data events for event data stores with the AWS CLI.

The following table lists the AWS IoT SiteWise resource types. The **Data event type (console)** column shows the value to choose from the **Data event type** list on the CloudTrail console. The **resources.type value** column shows the `resources.type` value, which you would specify when configuring advanced event selectors using the AWS CLI or CloudTrail APIs. The **Data APIs logged to CloudTrail** column shows the API calls logged to CloudTrail for the resource type.

Data event type (console)	resources.type value	Data APIs logged to CloudTrail*
AWS IoT SiteWise asset	`AWS::IoTSiteWise::Asset`	• BatchPutAssetPropertyValue • GetAssetPropertyValue • GetAssetPropertyValueHistory • GetAssetPropertyAggregates • GetInterpolatedAssetPropertyValues • BatchGetAssetPropertyValue • BatchGetAssetPropertyValueHistory • BatchGetAssetPropertyAggregates
AWS IoT SiteWise time series	`AWS::IoTSiteWise::TimeSeries`	• BatchPutAssetPropertyValue • GetAssetPropertyValue • GetAssetPropertyValueHistory • GetAssetPropertyAggregates • GetInterpolatedAssetPropertyValues • BatchGetAssetPropertyValue • BatchGetAssetPropertyValueHistory

Data event type (console)	resources.type value	Data APIs logged to CloudTrail*
		• BatchGetAssetPropertyAggregates

> **Note**
>
> The resources.type logged in the Cloudtrail event depends on the identifier used in the API request. If an asset id is specified in the request then the Asset resources.type is logged, else the TimeSeries resources.type is logged.

*You can configure advanced event selectors to filter on the `eventName`, `readOnly`, and `resources.ARN` fields to log only those events that are important to you. For more information about these fields, see AdvancedFieldSelector.

AWS IoT SiteWise management events in CloudTrail

Management events provide information about management operations that are performed on resources in your AWS account. These are also known as control plane operations. By default, CloudTrail logs management events.

AWS IoT SiteWise logs all AWS IoT SiteWise control plane operations as management events. For a list of the AWS IoT SiteWise control plane operations that AWS IoT SiteWise logs to CloudTrail, see the AWS IoT SiteWise API Reference.

Example: AWS IoT SiteWise log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested operation, the date and time of the operation, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `CreateAsset` operation.

```
{
```

```
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/Administrator",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "Administrator",
    "sessionContext": {
      "sessionIssuer": {},
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-03-11T17:26:40Z"
      }
    },
    "invokedBy": "signin.amazonaws.com"
  },
  "eventTime": "2020-03-11T18:01:22Z",
  "eventSource": "iotsitewise.amazonaws.com",
  "eventName": "CreateAsset",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "signin.amazonaws.com",
  "requestParameters": {
    "assetName": "Wind Turbine 1",
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "clientToken": "a1b2c3d4-5678-90ab-cdef-00000EXAMPLE"
  },
  "responseElements": {
    "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
    "assetArn": "arn:aws:iotsitewise:us-east-1:123456789012:asset/a1b2c3d4-5678-90ab-
cdef-22222EXAMPLE",
    "assetStatus": {
      "state": "CREATING"
    }
  },
  "requestID": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
  "eventID": "a1b2c3d4-5678-90ab-cdef-bbbbbEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Tagging your AWS IoT SiteWise resources

Tagging your AWS IoT SiteWise resources provides a powerful way to categorize, manage, and retrieve organizational assets efficiently. By assigning tags, which consist of key-value pairs, you can attach descriptive metadata to your resources. The metadata from tags can be used to streamline operations. For example, in a wind farm scenario, tags allow you to label turbines with specific attributes like location, capacity, and operational status, enabling quick identification and management within AWS IoT SiteWise.

Integrating tags with AWS Identity and Access Management (IAM) policies enhances security and operational control by defining conditional access rules. This means you can specify that only users with certain tags. For example, only those tagged with a certain role or department, can access or modify particular resources.

Using tags in AWS IoT SiteWise

Use tags to categorize your AWS IoT SiteWise resources by purpose, owner, environment, or any other classification for your use case. When you have many resources of the same type, you can quickly identify a specific resource based on its tags.

Each tag is made up of a key and an optional value that you specify. For example, you can establish a series of tags for your asset models to track them according to the industrial processes they support. It's recommended to develop a tailored set of tag keys for each type of resource you manage. Using a consistent set of tag keys can makes it easier manage resources.

Tagging with the AWS Management Console

The **Tag Editor** in the AWS Management Console provides a central, unified way for you to create and manage your tags for resources from all AWS services. For more information, see Tag Editor in the *AWS Resource Groups User Guide*.

Tagging with the AWS IoT SiteWise API

The AWS IoT SiteWise API also uses tags. Before you create tags, be aware of tagging restrictions. For more information, see Tag naming and usage conventions in the *AWS General Reference*.

- To add tags when you create a resource, define them in the `tags` property of the resource.

- To add tags to an existing resource, or to update tag values, use the TagResource operation.
- To remove tags from a resource, use the UntagResource operation.
- To retrieve the tags that are associated with a resource, use the ListTagsForResource operation, or describe the resource and inspect its `tags` property.

The following table lists resources you can tag using the AWS IoT SiteWise API and their corresponding `Create` and `Describe` operations.

Taggable AWS IoT SiteWise resources

Resource	Create operation	Describe operation
Asset model or component model	CreateAssetModel	DescribeAssetModel
Asset	CreateAsset	DescribeAsset
SiteWise Edge gateway	CreateGateway	DescribeGateway
Portal	CreatePortal	DescribePortal
Project	CreateProject	DescribeProject
Dashboard	CreateDashboard	DescribeDashboard
Access policy	CreateAccessPolicy	DescribeAccessPolicy
Time series	BatchPutAssetPropertyValue	DescribeTimeSeries

For `BatchPutAssetPropertyValue`, you can configure your data sources to send industrial data to AWS IoT SiteWise before you create asset models and assets. AWS IoT SiteWise automatically creates data streams to receive streams of raw data from your equipment. For more information, see Managing data ingestion.

Use the following operations to view and manage tags for resources that support tagging:

- TagResource – Adds tags to a resource, or updates an existing tag's value.
- ListTagsForResource – Lists the tags for a resource.
- UntagResource – Removes tags from a resource.

Add or remove tags from a resource at any time. To update the value of an existing tag key, add a new tag with the same key and your desired new value to the resource. This action replaces the old value with the new one. While it's possible to assign an empty string as a tag value, you can't assign a null value.

Deleting a resource also removes any tags linked to it.

Using tags with IAM policies

Use resource tags in your IAM policies to control user access and permissions. For example, policies can allow users to only create resources that have a specific tag attached. Policies can also restrict users from creating or modifying resources that have certain tags.

> **Note**
>
> If you use tags to allow or deny users' access to resources, you should deny users the ability to add or remove those tags for the same resources. Otherwise, a user could bypass your restrictions and gain access to a resource by modifying its tags.

You can use the following condition context keys and values in the `Condition` element (also called the `Condition` block) of a policy statement.

`aws:ResourceTag/`*`tag-key`*`: `*`tag-value`*

Allow or deny actions on resources with specific tags.

`aws:RequestTag/`*`tag-key`*`: `*`tag-value`*

Require that a specific tag be used (or not used) when creating or modifying a taggable resource.

`aws:TagKeys: [`*`tag-key`*`, ...]`

Require that a specific set of tag keys be used (or not used) when creating or modifying a taggable resource.

> **Note**
>
> The condition context keys and values in an IAM policy apply only to actions that have a taggable resource as a required parameter. For example, you can set tag-based conditional

access for ListAssets. You can't set tag-based conditional access on PutLoggingOptions because no taggable resource is referenced in the request.

For more information, see Controlling access to AWS resources using resource tags and IAM JSON policy reference in the *IAM User Guide*.

Example IAM policies using tags

- Viewing AWS IoT SiteWise assets based on tags

Troubleshooting AWS IoT SiteWise

Use the information in these sections to troubleshoot and resolve issues with AWS IoT SiteWise.

Topics

Troubleshooting bulk import and export operations

To handle and diagnose errors produced during a transfer job, see the AWS IoT TwinMaker **GetMetadataTransferJob** API:

1. After creating and running a transfer job, call the **GetMetadataTransferJob** API:

```
aws iottwinmaker get-metadata-transfer-job \
--metadata-transfer-job-id your_metadata_transfer_job_id \
--region us-east-1
```

2. The state of the job changes to one of the below states:
   - COMPLETED
   - CANCELLED
   - ERROR
3. The **GetMetadataTransferJob** API returns a MetadataTransferJobProgress object.
4. The **MetadataTransferJobProgress** object contains the following parameters:
   - **failedCount** : Indicates the count of assets that failed during the transfer process.
   - **skippedCount** : Indicates the count of assets that were skipped during the transfer process.
   - **succeededCount** : Indicates the count of assets that succeeded during the transfer process.
   - **totalCount** : Indicates the total count of assets involved in the transfer process.

5. Additionally a **reportUrl** element is returned by the API call, which contains a pre-signed URL. If your transfer job has errors that needs investigation, you can download a full error report at this URL.

Troubleshooting an AWS IoT SiteWise portal

Troubleshoot common issues with your AWS IoT SiteWise portals.

Users and administrators can't access AWS IoT SiteWise portal

If users or administrators cannot access your AWS IoT SiteWise portal, you may have restricted permissions in attached AWS Identity and Access Management (IAM) policies that prevent successful logins.

See the following examples of IAM policies that will result in login failure:

> **Note**
>
> Any attached IAM policies that include a `"Condition"` element will cause a login failure.

Example 1: The condition here is a limited IP, and this will cause a login failure.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:DescribePortal"
            ],
            "Resource": "*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "REPLACESAMPLEIP"
                    ]
                }
            }
        }
    ]
```

```
}
```

Example 2: The condition here is an included tag, and this will cause a login failure.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:DescribePortal"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/project": "*"
                }
            }
        }
    ]
}
```

When adding users or administrators to the portal, avoid creating IAM policies that restrict user permissions, such as limited IP. Any attached policies with restricted permissions will not be able to connect to the AWS IoT SiteWise portal.

Troubleshooting an SiteWise Edge gateway

AWS IoT SiteWise Edge gateways run a set of AWS IoT Greengrass components. You can configure your SiteWise Edge gateway to log events to Amazon CloudWatch and to your SiteWise Edge gateway's local file system. Then, you can view the log files to troubleshoot your SiteWise Edge gateway.

You can also view CloudWatch metrics reported by your SiteWise Edge gateways to troubleshoot issues with connectivity or data streams. For more information, see Monitoring AWS IoT SiteWise with Amazon CloudWatch metrics.

Topics

- Configuring and accessing SiteWise Edge gateway logs
- Troubleshooting SiteWise Edge gateway issues

- Troubleshooting AWS IoT Greengrass issues

Configuring and accessing SiteWise Edge gateway logs

Before you can view SiteWise Edge gateway logs, you must configure your SiteWise Edge gateway to send logs to Amazon CloudWatch Logs or store logs on the local file system.

- Use CloudWatch Logs if you want to use the AWS Management Console to view your SiteWise Edge gateway's log files. For more information, see Using Amazon CloudWatch Logs.
- Use local file system logs if you want to use the command line or local software to view your SiteWise Edge gateway's log files. For more information, see Using service logs.

Troubleshooting SiteWise Edge gateway issues

Use the following information to troubleshoot SiteWise Edge gateway issues.

Issues

- Unable to deploy packs to SiteWise Edge gateways
- AWS IoT SiteWise doesn't receive data from OPC-UA servers
- No data was shown in the dashboard
- "Could not find or load main class" showing up in the aws.iot.SiteWiseEdgePublisher logs at /greengrass/v2/logs error

Unable to deploy packs to SiteWise Edge gateways

If the AWS IoT Greengrass nucleus component (`aws.greengrass.Nucleus`) is out of date, you might not be able to deploy packs to your SiteWise Edge gateway. You can use the AWS IoT Greengrass V2 console to upgrade the AWS IoT Greengrass nucleus component.

Upgrade the AWS IoT Greengrass nucleus component (console)

1. Navigate to the AWS IoT Greengrass console.
2. In the navigation pane, under **AWS IoT Greengrass**, choose **Deployments**.
3. In the **Deployments** list, select the deployment that you want to revise.
4. Choose **Revise**.

5. On the **Specify target** page, choose **Next**.
6. On the **Select components** page, under **Public components**, in the search box, enter **`aws.greengrass.Nucleus`**, and then select **aws.greengrass.Nucleus**.
7. Choose **Next**.
8. On the **Configure components** page, choose **Next**.
9. On the **Configure advanced settings** page, choose **Next**.
10. On the **Review** page, choose **Deploy**.

AWS IoT SiteWise doesn't receive data from OPC-UA servers

If your AWS IoT SiteWise assets aren't receiving data sent by your OPC-UA servers, you can search your SiteWise Edge gateway's logs to troubleshoot issues. Look for info-level `swPublisher` logs that contain the following message.

```
Emitting diagnostic name=PublishError.SomeException
```

Based on the type of *`SomeException`* in the log, use the following exception types and corresponding issues to troubleshoot your SiteWise Edge gateway:

- **ResourceNotFoundException** – Your OPC-UA servers are sending data that doesn't match a property alias for any asset. This exception can occur in two cases:
  - Your property aliases don't exactly match your OPC-UA variables, including any source prefix you defined. Check that your property aliases and source prefixes are correct.
  - You haven't mapped your OPC-UA variables to asset properties. For more information, see Mapping industrial data streams to asset properties.

    If you already mapped all of the OPC-UA variables that you want in AWS IoT SiteWise, you can filter which OPC-UA variables the SiteWise Edge gateway sends. For more information, see Using OPC-UA node filters.
- **InvalidRequestException** – Your OPC-UA variables data types don't match your asset property data types. For example, if an OPC-UA variable has an integer data type, your corresponding asset property must be integer data type. A double-type asset property can't receive OPC-UA integer values. To fix this issue, define new properties with the correct data types.
- **TimestampOutOfRangeException** – Your SiteWise Edge gateway is sending data that is outside the range that AWS IoT SiteWise accepts. AWS IoT SiteWise rejects any data points with

timestamps earlier than 7 days in the past or newer than 5 minutes in the future. If your SiteWise Edge gateway lost power or connection to the AWS Cloud, you might need to clear your SiteWise Edge gateway's cache.

- **ThrottlingException** or **LimitExceededException** – Your request exceeded an AWS IoT SiteWise service quota, such as rate of data points ingested or request rate for asset property data API operations. Check that your configuration doesn't exceed the AWS IoT SiteWise quotas.

No data was shown in the dashboard

If there is no data shown in your dashboard, the **Publisher configuration** and the **Data Source** of the SiteWise Edge gateway may be out of sync. If they are out of sync, updating the name of the data source may expedite the sync from the cloud to the edge, fixing the Out of sync error.

To update the name of a data source

1. Navigate to the AWS IoT SiteWise console.
2. In the navigation pane, choose **Edge gateways**.
3. Select the SiteWise Edge gateway connected to the dashboard.
4. Under **Data sources**, select **Edit**.
5. Select a new source **Name**, and select **Save** to confirm your change.
6. Verify your changes by confirming the the data source name has been updated in the **Data sources** table.

"Could not find or load main class" showing up in the aws.iot.SiteWiseEdgePublisher logs at /greengrass/v2/logs error

If you see this error, you may need to update the java version of your SiteWise Edge gateway.

- From a terminal, run the following command:

```
java -version
```

The version of java your SiteWise Edge gateway is running with will show up under `OpenJDK Runtime Environment`. You'll see a response like the following:

```
openjdk version "11.0.20" 2023-07-18 LTS
```

```
OpenJDK Runtime Environment Corretto011.0.20.8.1 (build 11.0.20+8-LTS
OpenJDK 64-Bit Server VM Corretto-11.0.20.8.1 (build 11.0.20+8-LTS, mixed node)
```

If you are running Java version 11.0.20.8.1 you must update the IoT SiteWise Publisher pack to version 2.4.1 or newer. Only java version 11.0.20.8.1 is affected, environments with other java versions can continue to use older versions of the IoT SiteWise Publisher component. For more information about updating a component pack, see Changing the version of SiteWise Edge gateway component packs.

Troubleshooting AWS IoT Greengrass issues

To find solutions to many issues configuring or deploying your SiteWise Edge gateway on AWS IoT Greengrass, see Troubleshooting AWS IoT Greengrass in the *AWS IoT Greengrass Developer Guide*.

Troubleshooting an AWS IoT SiteWise rule action

To troubleshoot your AWS IoT SiteWise rule action in AWS IoT Core, you can do one of the following procedures:

- Configure Amazon CloudWatch Logs
- Configure a republish error action for your rule

Then, compare the error messages with the errors in this topic to troubleshoot your issue.

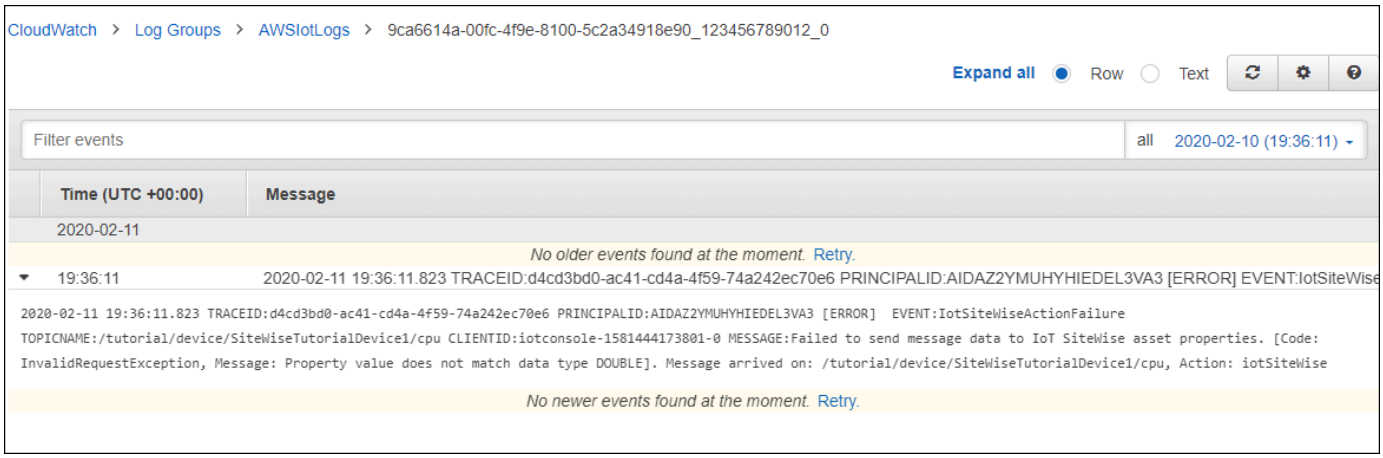
Topics

Configuring AWS IoT Core logs

You can configure AWS IoT to log various levels of information to CloudWatch Logs.

To configure and access CloudWatch Logs

1. To configure logging for AWS IoT Core, see Monitoring with CloudWatch Logs in the *AWS IoT Developer Guide*.
2. Navigate to the CloudWatch console.
3. In the navigation pane, choose **Log groups**.
4. Choose the **AWSIotLogs** group.
5. Choose a recent log stream. By default, CloudWatch displays the most recent log stream first.
6. Choose a log entry to expand the log message. Your log entry might look like the following screenshot.



7. Compare the error messages with the errors in this topic to troubleshoot your issue.

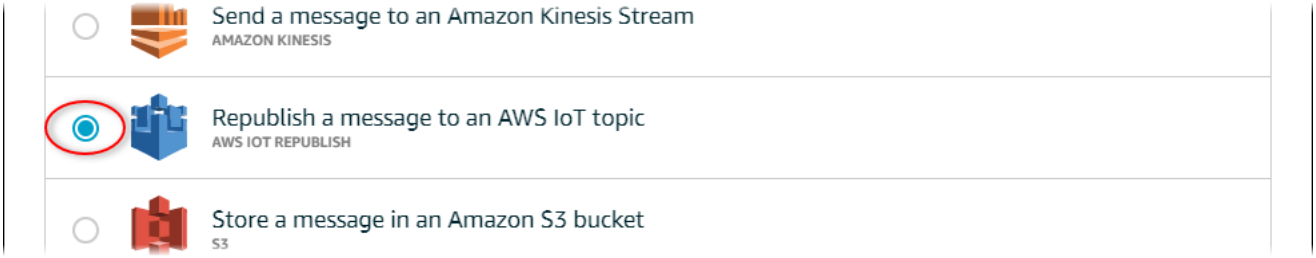
Configuring a republish error action

You can configure an error action on your rule to handle error messages. In this procedure, you configure the republish rule action as an error action to view error messages in the MQTT test client.

> **Note**
>
> The republish error action outputs only the equivalent of ERROR level logs. If you want more verbose logs, you must configure CloudWatch Logs.

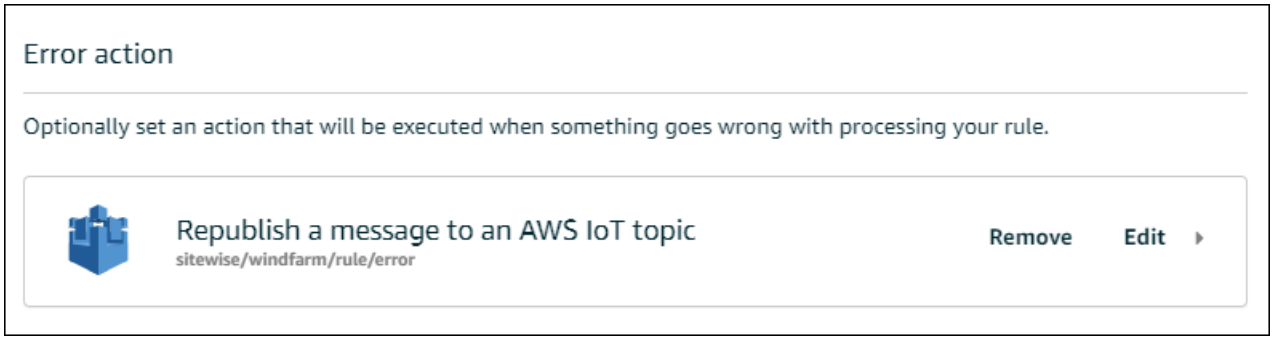
To add a republish error action to a rule

1. Navigate to the AWS IoT console.
2. In the left navigation pane, choose **Act** and then choose **Rules**.
3. Choose your rule.
4. Under **Error action**, choose **Add action**.
5. Choose **Republish a message to an AWS IoT topic**.



6. Choose **Configure action** at the bottom of the page.
7. In **Topic**, enter a unique topic (for example, **`sitewise/windfarm/rule/error`**). AWS IoT Core will republish error messages to this topic.
8. Choose **Select** to grant AWS IoT Core access to perform the error action.
9. Choose **Select** next to the role that you created for the rule.
10. Choose **Update Role** to add the additional permissions to the role.
11. Choose **Add action**.

    Your rule's error action should look similar to the following screenshot.



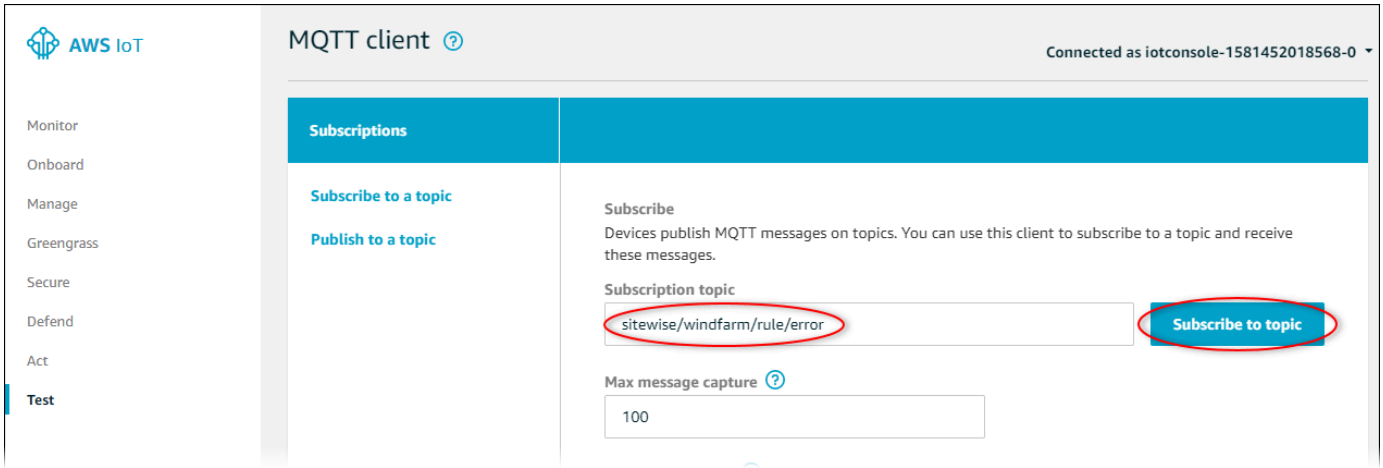
12. Choose the back arrow in the upper left of the console to return to the AWS IoT console home.

After you set up the republish error action, you can view the error messages in the MQTT test client in AWS IoT Core.

In the following procedure, you subscribe to the error topic in the MQTT test client. In the MQTT test client, you can receive your rule's error messages to troubleshoot the issue.

To subscribe to the error action topic

1. Navigate to the AWS IoT console.
2. In the left navigation page, choose **Test** to open the MQTT test client.
3. In the **Subscription topic** field, enter the error topic that you configured earlier (for example, **`sitewise/windfarm/rule/error`**) and choose **Subscribe to topic**.



4. Watch for error messages to appear and then expand the `failures` array in any error message.

   Next, compare the error messages with the errors in this topic to troubleshoot your issue.

Troubleshooting issues

Use the following information to troubleshoot rule issues.

Issues

- Error: Member must be within 604800 seconds before and 300 seconds after the current timestamp
- Error: Property value does not match data type <type>
- Error: User: <role-arn> is not authorized to perform: iotsitewise:BatchPutAssetPropertyValue on resource
- Error: iot.amazonaws.com is unable to perform: sts:AssumeRole on resource: <role-arn>

- Info: No requests were sent. PutAssetPropertyValueEntries was empty after performing substitution templates.

Error: Member must be within 604800 seconds before and 300 seconds after the current timestamp

Your timestamp is older than 7 days or newer than 5 minutes, compared to current Unix epoch time. Try the following:

- Check that your timestamp is in Unix epoch (UTC) time. If you provide a timestamp with a different timezone, you receive this error.
- Check that your timestamp is in seconds. AWS IoT SiteWise expects timestamps split into time in seconds (in Unix epoch time) and offset in nanoseconds.
- Check that you're uploading data that is timestamped no later than 7 days in the past.

Error: Property value does not match data type <type>

An entry in your rule action has a different data type than the target asset property. For example, your target asset property is a `DOUBLE` and your selected data type is **Integer** or you passed the value in `integerValue`. Try the following:

- If you configure the rule from the AWS IoT console, check that you chose the correct **Data type** for each entry.
- If you configure the rule from the API or AWS Command Line Interface (AWS CLI), check that your `value` object uses the correct type field (for example, `doubleValue` for a `DOUBLE` property).

Error: User: <role-arn> is not authorized to perform: iotsitewise:BatchPutAssetPropertyValue on resource

Your rule isn't authorized to access the target asset property, or the target asset property doesn't exist. Try the following:

- Check that your property alias is correct and that you have an asset property with the given property alias. For more information, see Mapping industrial data streams to asset properties.

- Check that your rule has a role and that the role allows `iotsitewise:BatchPutAssetPropertyValue` permission to the targeted asset property, such as through the target asset's hierarchy. For more information, see Granting AWS IoT the required access.

Error: iot.amazonaws.com is unable to perform: sts:AssumeRole on resource: <role-arn>

Your user isn't authorized to assume the role on your rule in AWS Identity and Access Management (IAM).

Check that your user is allowed `iam:PassRole` permission to the role on your rule. For more information, see Pass role permissions in the *AWS IoT Developer Guide*.

Info: No requests were sent. PutAssetPropertyValueEntries was empty after performing substitution templates.

> **Note**
>
> This message is an `INFO` level log.

Your request must have at least one entry with all of the required parameters.

Check that your rule's parameters, including substitution templates, result in non-empty values. Substitution templates can't access values defined in AS clauses in your rule query statement. For more information, see Substitution templates in the *AWS IoT Developer Guide*.

Troubleshooting a rule

Follow the steps in this procedure to troubleshoot your rule if the CPU and memory usage data isn't appearing in AWS IoT SiteWise as expected. In this procedure, you configure the republish rule action as an error action to view error messages in the MQTT test client. You can also configure logging to CloudWatch Logs to troubleshoot. For more information, see Troubleshooting an AWS IoT SiteWise rule action.

To add a republish error action to a rule

1. Navigate to the AWS IoT console.

2. In the left navigation pane, choose **Message routing** and then choose **Rules**.
3. Choose the rule that you created earlier and choose **Edit**.
4. Under **Error action - *optional***, choose **Add error action**.
5. Choose **Republish a message to an AWS IoT topic**.
6. In **Topic**, enter the path to your error (for example, **`sitewise/rule/tutorial/error`**). AWS IoT Core will republish error messages to this topic.
7. Choose the role that you created earlier (for example, **SiteWiseTutorialDeviceRuleRole**).
8. Choose **Update**.

After you set up the republish error action, you can view the error messages in the MQTT test client in AWS IoT Core.

In the following procedure, you subscribe to the error topic in the MQTT test client.

To subscribe to the error action topic

1. Navigate to the AWS IoT console.
2. In the left navigation page, choose **MQTT test client** to open the MQTT test client.
3. In the **Topic filter** field, enter **`sitewise/rule/tutorial/error`** and choose **Subscribe**.

When error messages appear, view the `failures` array in any error message to diagnose issues. For more information about possible issues and how to resolve them, see Troubleshooting an AWS IoT SiteWise rule action.

If errors don't appear, check that your rule is enabled and that you subscribed to the same topic that you configured in the republish error action. If errors still don't appear after you do that, check that the device script is running and updating the device's shadow successfully.

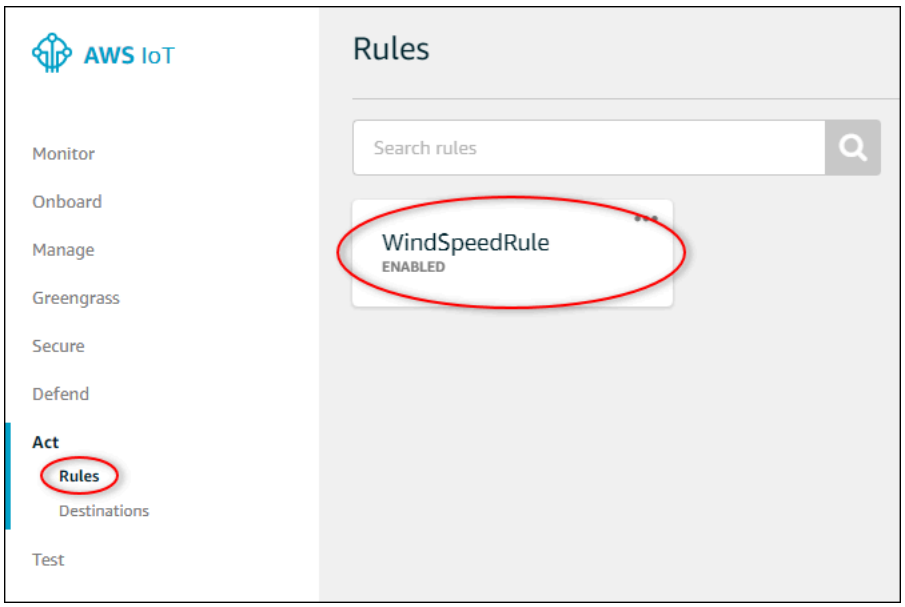
> **Note**
>
> You can also subscribe to your device's shadow update topic to view the payload that your AWS IoT SiteWise action parses. To do so, subscribe to the following topic.
>
> ```
> $aws/things/+/shadow/update/accepted
> ```

Troubleshooting a rule

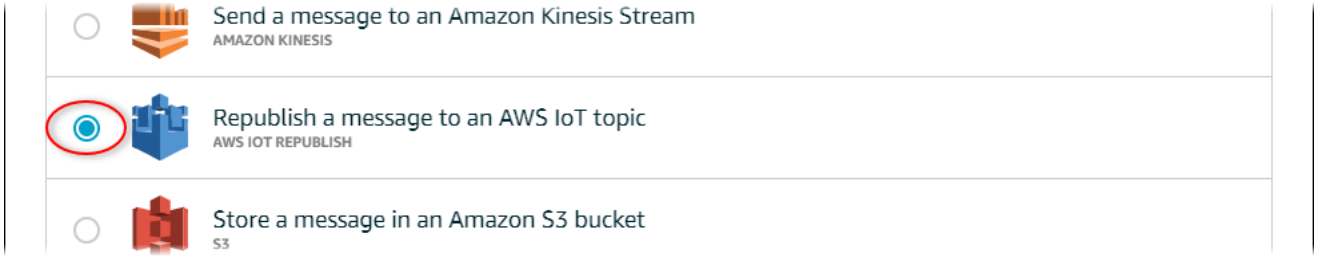
Follow the steps in this procedure to troubleshoot your rule if the demo asset data isn't appearing in the DynamoDB table as expected. In this procedure, you configure the republish rule action as an error action to view error messages in the MQTT test client. You can also configure logging to CloudWatch Logs to troubleshoot. For more information, see Monitoring with CloudWatch Logs in the *AWS IoT Developer Guide*.

To add a republish error action to a rule

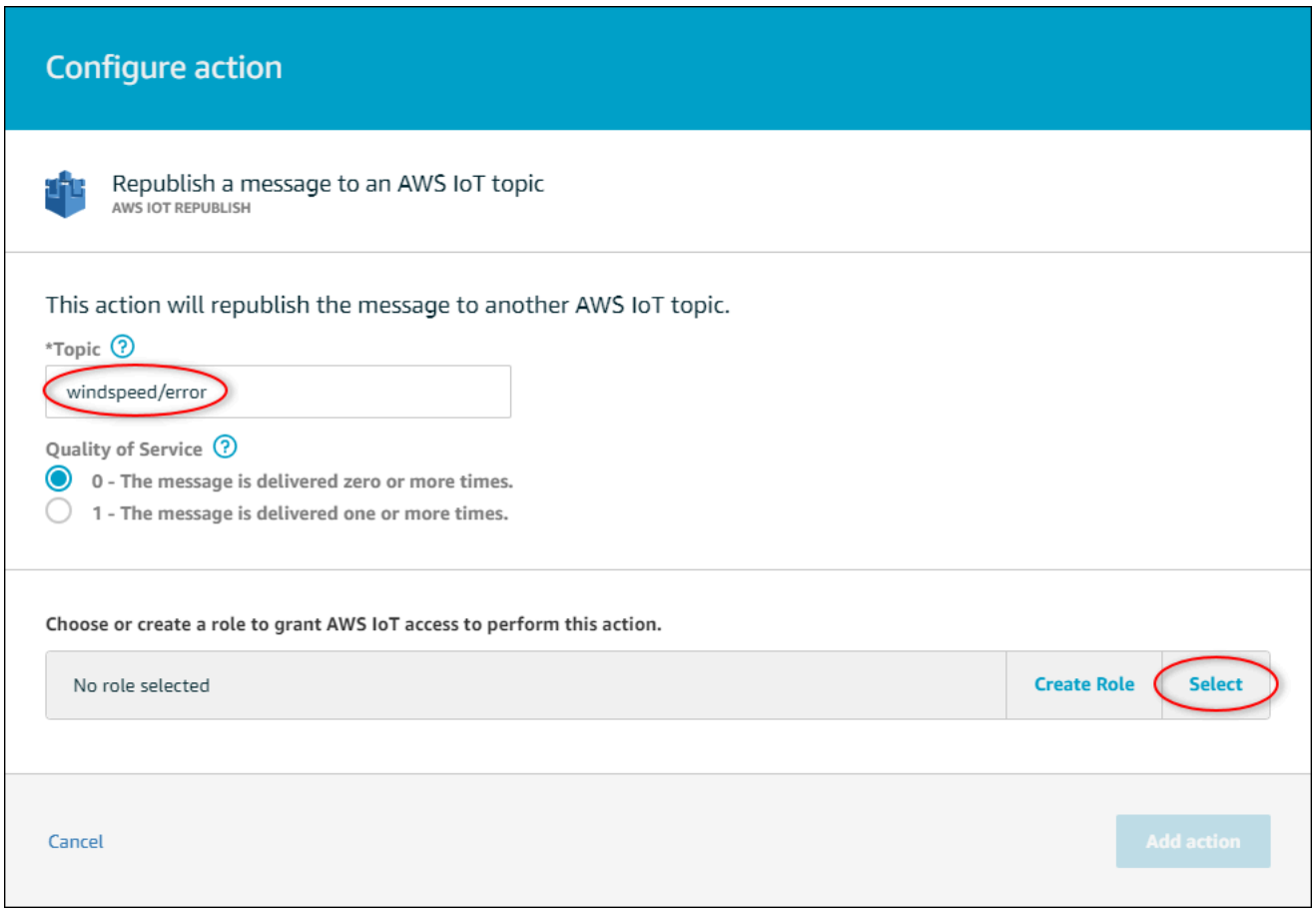
1. Navigate to the AWS IoT console.
2. In the left navigation pane, choose **Act** and then choose **Rules**.
3. Choose the rule that you created earlier.



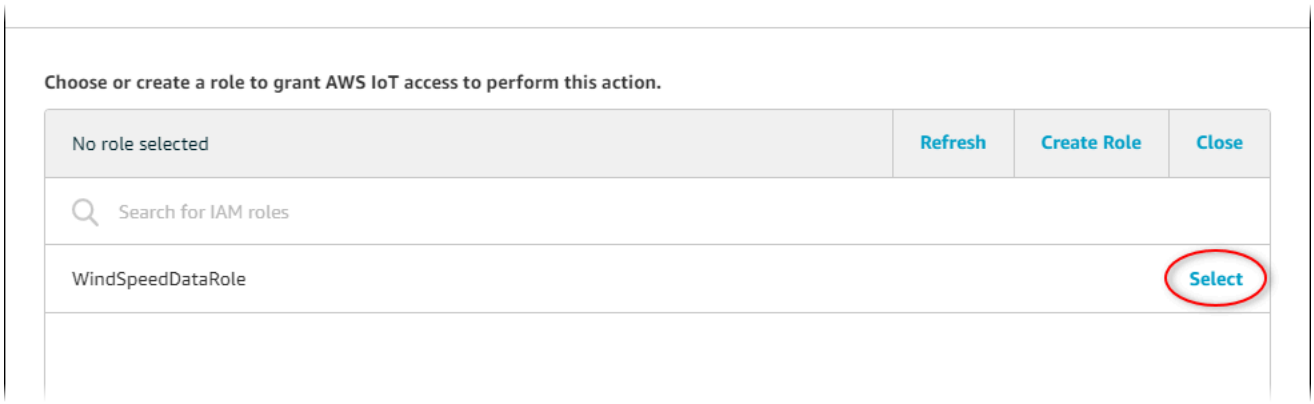
4. Under **Error action**, choose **Add action**.
5. Choose **Republish a message to an AWS IoT topic**.



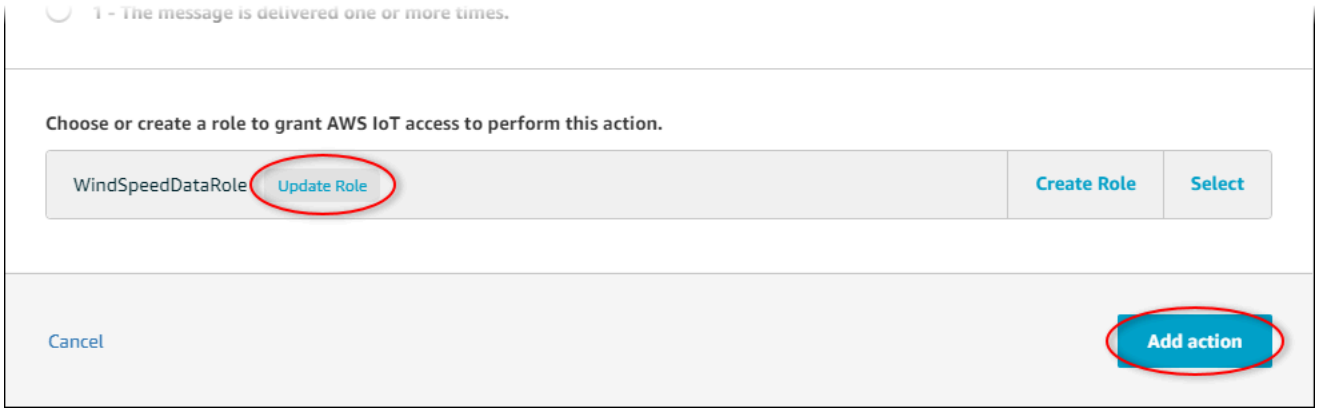
6. Choose **Configure action** at the bottom of the page.
7. In **Topic**, enter **`windspeed/error`**. AWS IoT Core will republish error messages to this topic.



8. Choose **Select** to grant AWS IoT Core access to perform the error action using the role that you created earlier.
9. Choose **Select** next to your role.



10. Choose **Update Role** to add the additional permissions to the role.



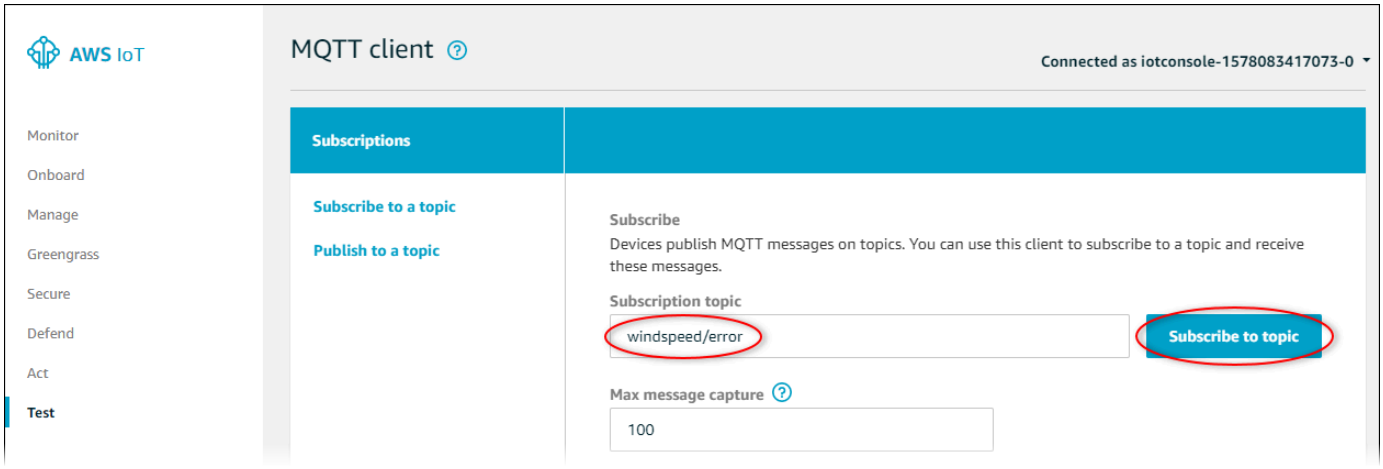
11. Choose **Add action** to finish adding the error action.
12. Choose the back arrow in the upper left of the console to return to the AWS IoT Core console home.

After you set up the republish error action, you can view the error messages in the MQTT test client in AWS IoT Core.

In the following procedure, you subscribe to the error topic in the MQTT test client.

To subscribe to the error action topic

1. In the AWS IoT Core console's left navigation page, choose **Test**.
2. In the **Subscription topic** field, enter **`windspeed/error`** and choose **Subscribe to topic**.



3. Watch for error messages to appear and explore the `failures` array in an error message to diagnose the following common issues:

   - Typos in the rule query statement
   - Insufficient role permissions

If errors don't appear, check that your rule is enabled and that you subscribed to the same topic that you configured in the republish error action. If errors still don't appear, check that your demo wind farm assets still exist and that you enabled notifications on the wind speed properties. If your demo assets expired and disappeared from AWS IoT SiteWise, you can create a new demo and update the rule query statement to reflect the updated asset model and property IDs.

AWS IoT SiteWise endpoints and quotas

The following sections describe the endpoints and quotas for AWS IoT SiteWise.

Contents

AWS IoT SiteWise endpoints

To connect programmatically to AWS IoT SiteWise, you use an endpoint. The AWS SDKs and the AWS Command Line Interface (AWS CLI) automatically use the default endpoint in an AWS Region. For more information about Regions where AWS IoT SiteWise is available, see AWS IoT SiteWise endpoints and quotas in the *AWS General Reference*.

AWS IoT SiteWise supports the following endpoints.

data.iotsitewise.region.amazonaws.com

Use this endpoint to access the following data plane API operations: BatchPutAssetPropertyValue, GetAssetPropertyAggregates, GetAssetPropertyValue, GetAssetPropertyValueHistory and GetInterpolatedAssetPropertyValues. Replace *region* with your AWS Region.

api.iotsitewise.region.amazonaws.com

AWS IoT SiteWise offers this consolidated endpoint for the control plane API operations that you use to manage asset models, assets, SiteWise Edge gateways, tags, and account configurations. Replace *region* with your AWS Region.

> **Note**
>
> - By default, AWS IoT SiteWise uses the consolidated endpoint when you make calls to the supported control plane API operations.
> - We recommend that you use the consolidated endpoint for the supported control plane API operations.
> - You can't use the consolidated endpoint to access the SiteWise Monitor API operations.

The supported control plane API operations include AssociateAssets, CreateAsset, CreateAssetModel, DeleteAsset, DeleteAssetModel, DeleteDashboard, DescribeAsset, DescribeAssetModel, DescribeAssetProperty, DescribeDashboard, DescribeLoggingOptions, DisassociateAssets, ListAssetModels, ListAssetRelationships, ListAssets, ListAssociatedAssets, PutLoggingOptions, UpdateAsset, UpdateAssetModel, UpdateAssetProperty, CreateGateway, DeleteGateway, DescribeGateway, DescribeGatewayCapabilityConfiguration, ListGateways, UpdateGateway, UpdateGatewayCapabilityConfiguration, DescribeStorageConfiguration, PutStorageConfiguration, DescribeDefaultEncryptionConfiguration, ListTagsForResource, PutDefaultEncryptionConfiguration, TagResource, and UntagResource.

The interface VPC endpoint for the control plane API operations only supports the consolidated endpoint. For more information, see VPC endpoints .

iotsitewise.region.amazonaws.com

Use this endpoint to access the following API operations: DescribeStorageConfiguration, PutStorageConfiguration, DescribeDefaultEncryptionConfiguration, ListTagsForResource, PutDefaultEncryptionConfiguration, TagResource, and UntagResource. Replace *region* with your AWS Region.

model.iotsitewise.region.amazonaws.com

Use this endpoint to access the following API operations: AssociateAssets, CreateAsset, CreateAssetModel, DeleteAsset, DeleteAssetModel, DeleteDashboard, DescribeAsset, DescribeAssetModel, DescribeAssetProperty, DescribeDashboard, DescribeLoggingOptions, DisassociateAssets, ListAssetModels, ListAssetRelationships, ListAssets, ListAssociatedAssets, PutLoggingOptions, UpdateAsset, UpdateAssetModel, and UpdateAssetProperty. Replace *region* with your AWS Region.

edge.iotsitewise.region.amazonaws.com

Use this endpoint to access the following API operations: CreateGateway, DeleteGateway, DescribeGateway, DescribeGatewayCapabilityConfiguration, ListGateways, UpdateGateway, and UpdateGatewayCapabilityConfiguration. Replace *region* with your AWS Region.

monitor.iotsitewise.region.amazonaws.com

Use this endpoint to access the following API operations: BatchAssociateProjectAssets, BatchDisassociateProjectAssets, CreateAccessPolicy, CreateDashboard, CreatePortal, CreateProject,

DeleteAccessPolicy, DeletePortal, DeleteProject, DescribeAccessPolicy, DescribePortal, DescribeProject, ListAccessPolicies, ListDashboards, ListPortals, ListProjectAssets, ListProjects, UpdateAccessPolicy, UpdateDashboard, UpdatePortal, and UpdateProject. Replace *region* with your AWS Region.

AWS IoT SiteWise quotas

The following tables describe quotas in AWS IoT SiteWise. For more information about quotas and how to request quota increases, see AWS service quotas in the *AWS General Reference*. For more information about AWS IoT SiteWise quotas, see AWS IoT SiteWise service quotas in the *AWS General Reference*.

Quotas for assets and asset models

Resource	Quota	Adjustable	Notes
Number of asset models per Region per AWS account	1000	Yes	
Number of assets per asset model	10,000	Yes	
Number of child assets per parent asset	2000	Yes	
Depth of asset model hierarchy tree	30	Yes	
Number of hierarchy definitions per asset model	30	Yes	
Number of properties in the root level per asset model	500	Yes	This maximum number of `assetModelProperties` for each asset model.

Resource	Quota	Adjustable	Notes
			This count does not include `compositeModelProperties`. This quota also applies to any unique asset created from this asset model.
Number of properties per asset model	5000	Yes	The maximum number of properties of an asset model of type `ASSET_MODEL` or `COMPONENT_MODEL`. This number is determined by combining the properties of the root asset model and any included component-model-based or inline composite models. This quota also applies to any unique asset created from this asset model.

Resource	Quota	Adjustable	Notes
Number of properties per composite model	100	Yes	The maximum number of properties allowed for composite models. Also, the maximum number of properties allowed for an asset model of type `COMPONENT_MODEL`.
Depth of property tree per asset model	10	No	For example, a model with a transform property C that consumes a transform property B that consumes a measurement property A has a depth of 3.
Number of asset models per hierarchy tree	100	Yes	

Resource	Quota	Adjustable	Notes
Number of directly dependent properties per asset model	20	No	This quota limits how many properties can directly depend on a single property, as defined in property formula expressio ns. The Number of dependent propertie s for an asset model must be greater than the Number of directly dependent properties per asset model. You must request a quota increase for both if the limit for the Number of directly dependent propertie s per asset model is greater than the limit for the Number of dependent properties per asset model.
Number of dependent properties per asset model	30	No	This quota limits how many properties can directly or indirectl y depend on a single property, as defined in property formula expressions.

Resource	Quota	Adjustable	Notes
Number of composite models per asset model	50	Yes	The maximum number of composite models allowed on a single asset model.
Composite model depth	2	Yes	The maximum depth of the composite model tree per asset model, including inline and component-model-based composite models.
Number of unique asset models that use the same component model	20	Yes	The maximum number of unique asset models that have at least one component-model-based composite model that directly references a specific asset model of type COMPONENT_MODEL.
Number of property variables per property formula expression	10	No	For example, there are two property variables, `power` and `temp`, in the expression `avg(power) + max(temp)`. This also applies for transform computation results.

Resource	Quota	Adjustable	Notes
Number of functions per property formula expression	10	No	For example, there are two functions , `avg` and `max`, in the expression `avg(power) + max(temp)` .

Quotas for asset property data

Resource	Quota	Adjustable	Notes
Request rate for asset property data API operations	1000 requests per second per Region per AWS account	Yes	This quota applies to API operations such as `GetAssetPropertyValue` and `BatchPutAssetPropertyValue` .
Number of data points per second per data quality per asset property	10 data points	No	This quota applies to the maximum number of timestamp-quality-value (TQV) data points with the same timestamp in seconds per data quality for each asset property. You can store up to this number of good-quality, uncertain-quality, and bad-quality data points for any given

Resource	Quota	Adjustable	Notes
			second for each asset property.
Number of `BatchPutAssetPropertyValue` entries ingested per second per asset property per Region per AWS Account.	10 entries per asset property	No	This quota applies to `BatchPutAssetPropertyValue` entries from all sources, including SiteWise Edge gateways, AWS IoT Core rules, and API calls.
Rate of data points ingested	5000 data points per second per Region per AWS account	Yes	Timestamp-quality-value (TQV) data points.
Request rate for `BatchGetAssetPropertyAggregates`	200	Yes	The maximum number of `BatchGetAssetPropertyAggregates` requests per second that you can perform in this account in the current Region.
Request rate for `BatchGetAssetPropertyValue`	500	Yes	The maximum number of `BatchGetAssetPropertyValue` requests per second that you can perform in this account in the current Region.

Resource	Quota	Adjustable	Notes
Request rate for `BatchGetAssetPropertyValueHistory`	200	Yes	The maximum number of `BatchGetAssetPropertyValueHistory` requests per second that you can perform in this account in the current Region.
Number of `BatchPutAssetPropertyValue` entries ingested per second per asset property per Region per AWS Account.	10 entries per asset property	No	This quota applies to `BatchPutAssetPropertyValue` entries from all sources, including SiteWise Edge gateways, AWS IoT Core rules, and API calls.
Rate of `GetAssetPropertyAggregates` requests and `BatchGetAssetPropertyAggregates` entry queries per asset property	50	No	The maximum number of total `GetAssetPropertyAggregates` requests and `BatchGetAssetPropertyAggregates` entries for each asset property per second in this account in the current Region.

Resource	Quota	Adjustable	Notes
Rate of `GetAssetPropertyValue` requests and `BatchGetAssetPropertyValue` entry queries per asset property	500	No	The maximum number of total `GetAssetPropertyValue` requests and `BatchGetAssetPropertyValue` entries for each asset property per second in this account in the current Region.
Rate of `GetAssetPropertyValueHistory` requests and `BatchGetAssetPropertyValueHistory` entry queries per asset property	30	No	The maximum number of total `GetAssetPropertyValueHistory` requests and `BatchGetAssetPropertyValueHistory` entries for each asset property per second in this account in the current Region.

Resource	Quota	Adjustable	Notes
Rate of `GetInterpolatedAssetPropertyValues` requests	500	Yes	The maximum number of `GetInterpolatedAssetPropertyValues` requests per second that you can perform in this account in the current Region.
Number of results per `GetInterpolatedAssetPropertyValues` request	10	Yes	The maximum number of results to return per paginated `GetInterpolatedAssetPropertyValues` request.

Resource	Quota	Adjustable	Notes
Rate of datapoints retrieved from `GetAssetPropertyValueHistory` and `BatchGetAssetPropertyValueHistory`	100MB read response per second per Region per AWS account.	Yes	The maximum byte rate (MB/second) of datapoints retrieved per second per Region per AWS account across `GetAssetPropertyValueHistory` and `BatchGetAssetPropertyValueHistory` . The response payload evaluated for this quota uses Timestamp-Quality-Value (TQV) fields for each datapoint and rounds the byte size for each API request to the next 4KB increment. Timestamp-quality-value (TQV) datapoints retrieved per second varies as per data type: • **Integer** – up to 5 Million TQV per second

Resource	Quota	Adjustable	Notes
			• **Double** – up to 4 Million TQV per second • **Boolean** – up to 6 Million TQV per second • **String** – varies based on each string value size.

Quotas for SiteWise Edge gateways

Resource	Quota	Adjustable
Number of SiteWise Edge gateways per Region per AWS account	100	Yes
Number of OPC-UA sources per SiteWise Edge gateway	100	No

Quotas for AWS IoT SiteWise Monitor

Resource	Quota	Adjustable
Number of portals per Region per AWS account	100	Yes
Number of projects per portal	100	Yes
Number of dashboards per project	100	Yes
Number of root assets per project	1	No

Resource	Quota	Adjustable
Number of visualizations per dashboard	10	Yes
Number of metrics per dashboard visualization	5	Yes
Number of thresholds per dashboard visualization	12	No

Quotas for AWS IoT SiteWise bulk import and export of metadata

Resource	Description	Quota	Adjustable
Number of metadata transfer jobs in queue	The maximum number of `PENDING` metadata transfer jobs in the queue.	10	Yes
Size of the metadata transfer job import file	The maximum size of the imported file (in MB).	100 MB	Yes
AWS IoT SiteWise resource quota for a metadata transfer job	The maximum number of resources imported or exported in a single job. A resource includes assets, and asset models.	5000	No

Quotas for AWS IoT SiteWise bulk import of data

Resource	Quota	Adjustable
Number of running bulk import jobs	100	No
Size of the CSV file	10 GB	No
Size of the uncompressed parquet file	256 MB	No
Size of the CSV file for buffered ingestion	256 MB	No
Size of the uncompressed parquet row group	64 MB	No
Number of unique measurements per parquet row group	2000	Yes
Number of days between the timestamp in the past and today for buffered ingestion	30	Yes
Request rate for `CreateBulkImportJobs` for each Region in each AWS account	10	Yes
Request rate for `ListBulkImportJobs` for each Region in each AWS account	50	Yes
Request rate for `DescribeBulkImportJobs` for each Region in each AWS account	50	Yes

Quotas for anomaly detection

The quotas for anomaly detection are shared between AWS IoT SiteWise and Amazon Lookout for Equipment. For more information, see Quotas for using Lookout for Equipment.

Document history for the AWS IoT SiteWise User Guide

The following table describes the documentation for this release of AWS IoT SiteWise.

- **API version:** 2019-12-02

Change	Description	Date
Added support for running SiteWise Edge on Siemens Industrial Edge	AWS IoT SiteWise now supports running SiteWise Edge on Siemens Industrial Edge devices.	November 26, 2023
Added support for warm tier storage	AWS IoT SiteWise now supports warm storage, a fully-managed storage tier that makes it easy for customers to securely store and access industrial data.	November 15, 2023
Added support for user-defined unique identifiers	AWS IoT SiteWise now supports the use of user-defined unique identifiers for asset, asset models, properties and hierarchies.	November 15, 2023
Added support for multivariate anomaly detection of industrial assets	AWS IoT SiteWise now supports multivariate anomaly detection of industrial assets by integration of historical and real time equipment data with Amazon Lookout for Equipment.	November 15, 2023
Added support for cost-efficient and scalable ingestion	AWS IoT SiteWise now supports cost-efficient and scalable ingestion of time-seri	November 15, 2023

of time-series data in AWS IoT SiteWise	es data needed for analytical use cases.	
Added support for bulk import, export, and update	AWS IoT SiteWise now supports bulk import, export, and update industrial equipment metadata.	November 15, 2023
Added support for asset model components	AWS IoT SiteWise now supports Asset model components to help industria l customers create reusable components.	November 15, 2023
Added support for IoT dashboard application	AWS IoT SiteWise now supports an open source dashboard application where you can visualize and interact with operational data.	November 15, 2023
Updated the service-linked roles for AWS IoT SiteWise	AWS IoT SiteWise has new service-linked roles, and can run a metadata search query, against the AWS IoT TwinMaker database.	November 6, 2023
Updated tagging for AWS IoT SiteWise data stream resources	Added support for tagging data stream resources.	August 18, 2022
Updated SiteWise Edge gateways	You can now configure the publisher to control what data is sent from the edge to the cloud and the order that it's sent to the cloud.	January 12, 2022

Updated the AWS IoT SiteWise demo	You can now use the demo to create a SiteWise Monitor portal.	January 10, 2022
Updated storage management	You can now define a retention period to control how long your data is kept in the hot tier.	November 29, 2021
Added support for data stream management	You can now ingest data to AWS IoT SiteWise before you create asset models and assets.	November 24, 2021
Updated asset model hierarchies	A child asset model now can be associated with multiple parent asset models.	October 28, 2021
Region launch	Launched AWS IoT SiteWise in AWS GovCloud (US-West).	September 29, 2021
Updated functions	Added the following features • In metrics, you can use nested expressions in aggregation functions and temporal functions. • In transforms, you can use the pretrigger() function to retrieve the value of a variable prior to the property update that triggered the current transform calculation.	August 10, 2021
Custom metric time interval	Added support for custom time intervals and offsets in metrics.	August 3, 2021

Using AWS IoT SiteWise at the edge	The edge processing feature is now generally available.	July 29, 2021
Exporting data to Amazon S3	AWS IoT SiteWise now can export data to Amazon S3.	July 27, 2021
VPC endpoints (AWS PrivateLink)	The interface VPC endpoint for the control plane API operations is now generally available.	July 15, 2021
Transforms	Transforms now can input multiple asset property variables.	July 8, 2021
Updated the timestamp() function	In transforms, you can now provide a variable as an argument to the `timestamp()` function.	June 16, 2021
Alarms general availability	The alarms feature is now generally available.	May 27, 2021
Modbus-TCP Protocol Adapter version 2 released	Version 2 of the Modbus-TCP Protocol Adapter connector is available. This release added support for ASCII, UTF8, and ISO8859 encoded source strings.	May 24, 2021

Updated service quotas	Added the following quotas for the GetInterpolatedAssetPropertyValues API: rate of `GetInterpolatedAssetPropertyValues` requests, number of results per `GetInterpolatedAssetPropertyValues` request, and number of days between the start date in the past and today for `GetInterpolatedAssetPropertyValues` .	April 29, 2021
Updated formula expressions	Added the following operators and functions: • Added the following operators: <, >, <=, >=, ==, `!=`, `!`, and, `or`, and `not`. • Added the following comparison function: `neq(x, y)`. • Added the following string functions: `join()`, `format()`, and `f''`.	April 22, 2021
VPC endpoints (AWS PrivateLink)	Added information about how to establish a private connection between your virtual private cloud (VPC) and the AWS IoT SiteWise control plane APIs by creating an interface VPC endpoint.	March 16, 2021

IAM federation	Your SiteWise Monitor portal administrators and users can now log in to their assigned portals with their IAM credentials.	March 16, 2021
Region launch	Launched AWS IoT SiteWise in China (Beijing).	February 3, 2021
IoT SiteWise connector version 10 released	Version 10 of the IoT SiteWise connector is available. This release configures `StreamManager` to improve handling when the source connection is lost and re-established. This version also accepts OPC-UA values with a `ServerTimestamp` when no `SourceTimestamp` is available.	January 22, 2021
Date and time functions	AWS IoT SiteWise now supports date and time functions.	January 21, 2021
Function syntax	You can now use Uniform Function Call Syntax (UFCS) for AWS IoT SiteWise functions.	January 11, 2021
Integrating with Grafana	Added information about how to visualize AWS IoT SiteWise data in Grafana dashboards.	December 15, 2020

AWS IoT SiteWise feature release	You can now monitor your data with alarms, process industrial data at the edge, use Modbus TCP and Ethernet/IP sources to your SiteWise Edge gateway, filter incoming data with deadbands, and more. • Added the Monitoring data with alarms section that you can use to define, configure, and respond to alarms in AWS IoT SiteWise. • Added the Edge processing section that you can use to configure processing of your industrial data on your edge devices. • Added the Modbus TCP and Ethernet/IP sections to the SiteWise Edge gateway source documentation. • Added the source destination section that you can use to customize where you send your incoming industrial data. • Added the OPC-UA filtering section that you can use to control the frequency and type of data that is sent to your SiteWise Edge gateway from your industrial local server.	December 15, 2020

AWS IoT SiteWise now supports customer managed CMKs.	AWS IoT SiteWise now supports encryption with customer managed CMKs.	November 24, 2020
IoT SiteWise connector version 8 released	Version 8 of the IoT SiteWise connector is available. This release improves stability when the connector experienc es intermittent network connectivity.	November 19, 2020
Using strings and conditionals in formula expressions	Added information about how to strings and conditional functions in formula expressio ns for transforms and metrics.	November 16, 2020
Ingesting data using AWS IoT Greengrass stream manager	Added information about how to ingest high-volume IoT data from local data sources using an AWS IoT Greengrass edge device.	September 16, 2020
VPC endpoints (AWS PrivateLi nk)	Added information about how to establish a private connection between your virtual private cloud (VPC) and the AWS IoT SiteWise data APIs by creating an interface VPC endpoint.	September 4, 2020
IoT SiteWise connector version 7 released	Version 7 of the IoT SiteWise connector is available. This release fixes an issue with SiteWise Edge gateway metrics.	August 14, 2020

Creating IAM Identity Center users from the AWS IoT SiteWise console	Added information about how you can create IAM Identity Center users in the AWS IoT SiteWise console. You can now create IAM Identity Center users when you assign users to a new or existing portal. Updated the Visualizing and sharing wind farm data tutorial to use this feature. This change reduces the number of steps in the tutorial.	August 4, 2020
Improved SiteWise Edge gateway troubleshooting	Added additional information about how to troubleshoot a SiteWise Edge gateway and how to export the OPC-UA client certificate for a source.	June 18, 2020
Console task documentation	Added console task documentation for Modeling industrial assets, Querying asset property data, and Interacting with other services. You can follow these instructions to complete tasks in the AWS IoT SiteWise console.	June 11, 2020
Analyzing exported data tutorial	Added a tutorial that you can follow to learn how to use Amazon Athena to analyze asset data that you exported to S3 with the export feature AWS CloudFormation template.	May 27, 2020

Improved using formula expressions	Added detailed informati on about the behavior of AWS IoT SiteWise formula properties and added an example of how to count filtered data points.	May 18, 2020
IoT SiteWise connector version 6 released	Version 6 of the IoT SiteWise connector is available. This release adds support for CloudWatch metrics and automatic discovery of new OPC-UA tags. This means you don't need to restart your SiteWise Edge gateway when tags change for your OPC-UA sources. This version of the connector requires stream manager and AWS IoT Greengrass Core software v1.10.0 or higher.	April 29, 2020

AWS IoT SiteWise feature release	AWS IoT SiteWise feature release. You can now manage SiteWise Edge gateways with the API, add your logo to portals, view SiteWise Edge gateway metrics, and more. • Added the Exporting data to Amazon S3 section with an AWS CloudFormation template that you can use to export new data values to an S3 bucket. • Added the Configuring data sources section that improves SiteWise Edge gateway source documentation and includes the new SiteWise Edge gateway APIs. • Added the SiteWise Edge gateway metrics section that describes the CloudWatch metrics that SiteWise Edge gateways publish. • Added the Configuring an SiteWise Edge gateway on Amazon EC2 section with an AWS CloudFormation template that you can use to quickly configure SiteWise Edge gateway dependencies on an Amazon EC2 instance.	April 29, 2020

	• Added the portal service roles section that describes the new permissions feature of SiteWise Monitor portals. • Updated portal documentation for portal service roles and portal logos. • Added the Tagging your AWS IoT SiteWise resources section. • Updated the Creating dashboards (CLI) section for the new dashboard definition structure. • Added the Security section.	
Ingesting data from AWS IoT Events	Added information about how to ingest data from AWS IoT Events when an event occurs.	April 20, 2020
Visualizing and sharing wind farm data in SiteWise Monitor tutorial	Added a tutorial that you can follow to learn how to use AWS IoT SiteWise Monitor to visualize and share asset data.	March 12, 2020
AWS IoT SiteWise concepts	Added a glossary of AWS IoT SiteWise concepts that you can use to learn about the service and its common terms.	March 5, 2020

Removed AWS IoT Greengrass installation instructions	Removed the AWS IoT Greengrass Core software installation instructions from the AWS IoT SiteWise User Guide. The AWS IoT Greengrass Developer Guide offers a device setup script and instructions to set up AWS IoT Greengrass on other platforms such as Amazon EC2 and Docker.	February 14, 2020
Improved ingesting data using AWS IoT Core rules	Added detailed informati on about how to use and how to troubleshoot the AWS IoT SiteWise rule action, which you can use to ingest data from MQTT messages through AWS IoT Core.	February 14, 2020
IoT SiteWise connector version 5 released	Version 5 of the IoT SiteWise connector is available. This release fixes a compatibility issue with AWS IoT Greengras s Core software v1.9.4.	February 12, 2020
IoT SiteWise connector version 4 released	Version 4 of the IoT SiteWise connector is available. This release fixes an issue with OPC-UA server reconnection.	February 7, 2020

Restructured modeling industrial assets	Restructured the Updating Assets and Models section into multiple topics within Modeling Industrial Assets. • Asset and model states • Mapping industrial data streams to asset properties • Updating attribute values • Associating and disassociating assets • Updating assets and models • Deleting assets and models	February 4, 2020
Ingesting data from AWS IoT things tutorial	Added a tutorial that you can follow to learn how to configure an AWS IoT SiteWise rule action to ingest data from a new or existing fleet of AWS IoT things.	February 4, 2020
Restructured retrieving data from AWS IoT SiteWise	Restructured the Retrieving Data section into two top-level sections: Querying asset property values and aggregates and Interacting with other AWS services.	January 21, 2020
Publishing property value updates to Amazon DynamoDB tutorial	Added a tutorial that you can follow to learn how to use property value notifications to store asset data in DynamoDB.	January 8, 2020

Using formula expressions	Added the formula expression reference to organize the constants and functions available for use in transform and metric properties. Restructured Asset properties into separate topics for each property type.	January 7, 2020
Using OPC-UA node filters	Added information about how to use OPC-UA node filters to improve SiteWise Edge gateway performance when adding SiteWise Edge gateway sources.	January 3, 2020
Upgrading a connector	Added information about how to upgrade a SiteWise Edge gateway when a new connector version is released.	December 30, 2019
IoT SiteWise connector version 3 released	Version 3 of the IoT SiteWise connector is available. This release removes the iot:* permissions requirement.	December 17, 2019
IoT SiteWise connector version 2 released	Version 2 of the IoT SiteWise connector is available. This release adds support for multiple OPC-UA secret resources.	December 10, 2019
Creating dashboards (AWS CLI)	Added information about how to create a dashboard in AWS IoT SiteWise Monitor using the AWS CLI.	December 6, 2019

AWS IoT SiteWise version 2 released	Released preview for version 2 of AWS IoT SiteWise. You can now ingest data over OPC-UA, MQTT, and HTTP, model your data in asset hierarchies, and visualize your data with SiteWise Monitor. • Rewrote the asset modeling section for changes to assets, asset models, and asset hierarchies. • Updated the data ingestion section to include AWS IoT Greengrass connector steps and non-gateway data ingestion sections. • Added the AWS IoT SiteWise Monitor section and a separate application guide that shows how to use the SiteWise Monitor web application. • Added Query data from AWS IoT SiteWise and Interacting with other AWS services sections. • Rewrote the getting started section to match the updated demo experience.	December 2, 2019
AWS IoT SiteWise version 1 released	Released initial preview for version 1 of AWS IoT SiteWise.	February 25, 2019

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.