AWS IoT

Developer Guide

AWS IoT: Developer Guide

Table of Contents

What Is AWS IoT?

AWS IoT provides secure, bi-directional communication between Internet-connected *things* (devices such as sensors, actuators, embedded micro-controllers, or smart appliances) and the AWS Cloud. This enables you to collect telemetry data from multiple things, and store and analyze the data. You can also create applications that enable your users to control these devices from their phones or tablets.

AWS IoT Components

AWS IoT consists of the following components:

Device gateway

Enables devices to securely and efficiently communicate with AWS IoT.

Message broker

Provides a secure mechanism for things and AWS IoT applications to publish and receive messages from each other. You can use either the MQTT protocol directly or MQTT over WebSocket to publish and subscribe. You can use the HTTP REST interface to publish.

Rules engine

Provides message processing and integration with other AWS services. You can use an SQL-based language to select data from message payloads, and then process and send the data to other services, such as Amazon S3, Amazon DynamoDB, and AWS Lambda. You can also use the message broker to republish messages to other subscribers.

Security and Identity service

Provides shared responsibility for security in the AWS Cloud. Your things must keep their credentials safe in order to securely send data to the message broker. The message broker and rules engine use AWS security features to send data securely to devices or other AWS services.

Thing registry

Sometimes referred to as the *device registry*. Organizes the resources associated with each thing. You register your things and associate up to three custom attributes with each thing. You can also associate certificates and MQTT client IDs with each thing to improve your ability to manage and troubleshoot your things.

Group registry

Thing groups allow you to manage several things at once by categorizing them into groups. Groups can also contain groups—you can build a hierarchy of groups. Any action you perform on a parent group will apply to its child groups, and to all the things in it and in all of its child groups as well. Permissions given to a group will apply to all things in the group and in all of its child groups.

Thing shadow

Sometimes referred to as a *device shadow*. A JSON document used to store and retrieve current state information for a thing (device, app, and so on).

Thing Shadows service

Provides persistent representations of your things in the AWS Cloud. You can publish updated state information to a thing shadow, and your thing can synchronize its state when it connects. Your things can also publish their current state to a thing shadow for use by applications or devices.

Device Provisioning service

Allows you to provision devices using a template that describes the resources required for your device: a thing, a certificate, and one or more policies. A thing is an entry in the device registry that

contains attributes that describe a device. Devices use certificates to authenticate with AWS IoT. Policies determine which operations a device can perform in AWS IoT.

The templates contain variables that are replaced by values in a dictionary (map). You can use the same template to provision multiple devices just by passing in different values for the template variables in the dictionary.

Custom Authentication service

You can define custom authorizers that allow you to manage your own authentication and authorization strategy using a custom authentication service and a Lambda function. Custom authorizers allow AWS IoT to authenticate your devices and authorize operations using bearer token authentication and authorization strategies.

Custom authorizers can implement various authentication strategies (for example: JWT verification, OAuth provider call out, and so on) and must return policy documents which are used by the device gateway to authorize MQTT operations.

Jobs Service

Allows you to define a set of remote operations that are sent to and executed on one or more devices connected to AWS IoT. For example, you can define a job that instructs a set of things to download and install application or firmware updates, reboot, rotate certificates, or perform remote troubleshooting operations.

To create a job, you specify a description of the remote operations to be performed and a list of targets that should perform them. The targets can be individual things, thing groups or both.

For information about AWS IoT limits, see AWS IoT Limits.

How to Get Started with AWS IoT

- To learn more about AWS IoT, see How AWS IoT Works (p. 3).
- To learn how to connect a thing to AWS IoT, see Getting Started with AWS IoT (p. 5).

Accessing AWS IoT

AWS IoT provides the following interfaces to create and interact with your things:

- **AWS Command Line Interface (AWS CLI)**—Run commands for AWS IoT on Windows, macOS, and Linux. These commands allow you to create and manage things, certificates, rules, and policies. To get started, see the AWS Command Line Interface User Guide. For more information about the commands for AWS IoT, see iot in the *AWS Command Line Interface Reference*.
- **AWS IoT API**—Build your IoT applications using HTTP or HTTPS requests. These API actions allow you to programmatically create and manage things, certificates, rules, and policies. For more information about the API actions for AWS IoT, see Actions in the *AWS IoT API Reference*.
- **AWS SDKs**—Build your IoT applications using language-specific APIs. These SDKs wrap the HTTP/HTTPS API and allow you to program in any of the supported languages. For more information, see AWS SDKs and Tools.
- **AWS IoT Device SDKs**—Build applications that run on devices that send messages to and receive messages from AWS IoT. For more information see, AWS IoT SDKs

Related Services

AWS IoT integrates directly with the following AWS services:

- **Amazon Simple Storage Service**—Provides scalable storage in the AWS Cloud. For more information, see Amazon S3.
- **Amazon DynamoDB**—Provides managed NoSQL databases. For more information, see Amazon DynamoDB.
- **Amazon Kinesis**—Enables real-time processing of streaming data at a massive scale. For more information, see Amazon Kinesis.
- **AWS Lambda**—Runs your code on virtual servers from Amazon EC2 in response to events. For more information, see AWS Lambda.
- **Amazon Simple Notification Service**—Sends or receives notifications. For more information, see Amazon SNS.
- **Amazon Simple Queue Service**—Stores data in a queue to be retrieved by applications. For more information, see Amazon SQS.

How AWS IoT Works

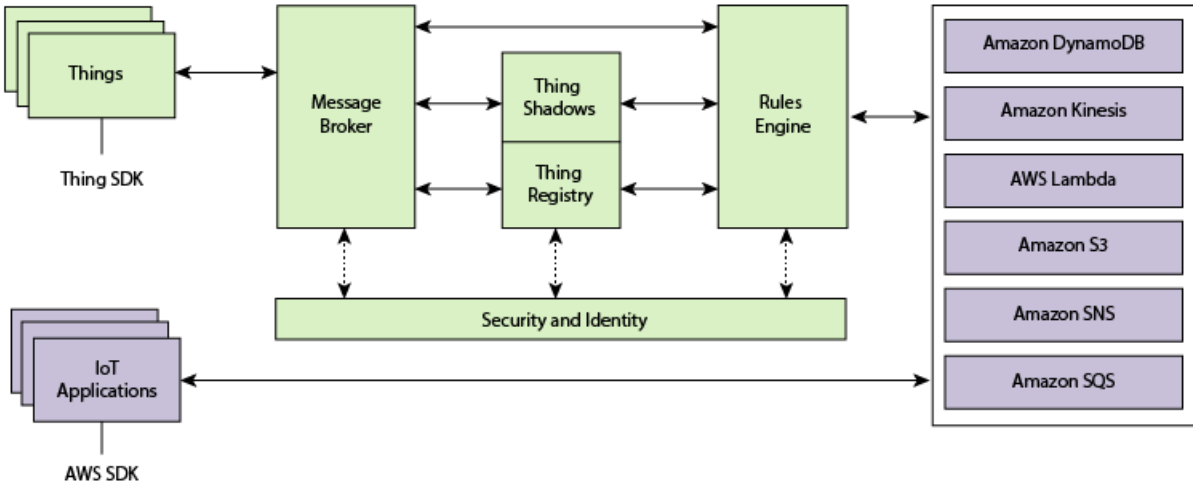
AWS IoT enables Internet-connected things to connect to the AWS Cloud and lets applications in the cloud interact with Internet-connected things. Common IoT applications either collect and process telemetry from devices or enable users to control a device remotely.

Things report their state by publishing messages, in JSON format, on MQTT topics. Each MQTT topic has a hierarchical name that identifies the thing whose state is being updated. When a message is published on an MQTT topic, the message is sent to the AWS IoT MQTT message broker, which is responsible for sending all messages published on an MQTT topic to all clients subscribed to that topic.

Communication between a thing and AWS IoT is protected through the use of X.509 certificates. AWS IoT can generate a certificate for you or you can use your own. In either case, the certificate must be registered and activated with AWS IoT, and then copied onto your thing. When your thing communicates with AWS IoT, it presents the certificate to AWS IoT as a credential.

We recommend that all things that connect to AWS IoT have an entry in the thing registry. The thing registry stores information about a thing and the certificates that are used by the thing to secure communication with AWS IoT.

You can create rules that define one or more actions to perform based on the data in a message. For example, you can insert, update, or query a DynamoDB table or invoke a Lambda function. Rules use expressions to filter messages. When a rule matches a message, the rules engine invokes the action using the selected properties. Rules also contain an IAM role that grants AWS IoT permission to the AWS resources used to perform the action.



Each thing has a thing shadow that stores and retrieves state information. Each item in the state information has two entries: the state last reported by the thing and the desired state requested by an application. An application can request the current state information for a thing. The shadow responds to the request by providing a JSON document with the state information (both reported and desired), metadata, and a version number. An application can control a thing by requesting a change in its state. The shadow accepts the state change request, updates its state information, and sends a message to indicate the state information has been updated. The thing receives the message, changes its state, and then reports its new state.

Getting Started with AWS IoT

This tutorial shows you how to create resources required to send, receive, and process MQTT messages from devices using AWS IoT.

You need the following to complete this tutorial:

- A computer with Wi-Fi access.
- If you have an AWS IoT button (pictured here), you can use it to complete this tutorial.
- If you do not have a button, you can purchase one here or you can use the MQTT client in the AWS IoT console to emulate a device.

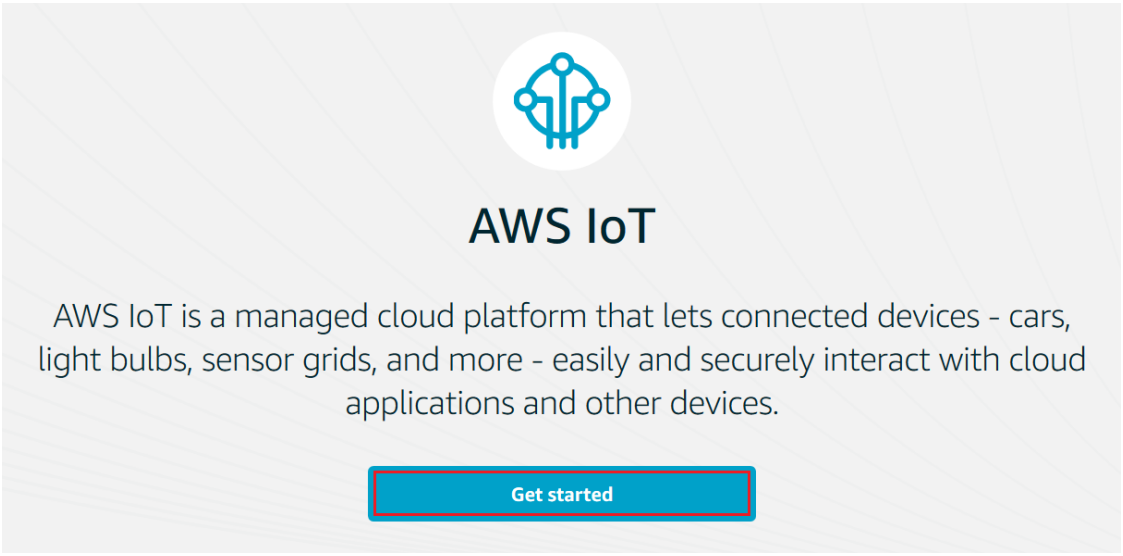
For more information about AWS IoT, see What Is AWS IoT (p. 1).

Sign in to the AWS IoT Console

If you do not have an AWS account, create one.

To create an AWS account:

1. Open the AWS home page and choose **Create an AWS Account**.
2. Follow the online instructions. Part of the sign-up procedure involves receiving a phone call and entering a PIN using your phone's keypad.
3. Sign in to the AWS Management Console and open the AWS IoT console.
4. On the **Welcome** page, choose **Get started**.



If this is your first time using the AWS IoT console, you see the **Welcome to the AWS IoT Console** page.

Register a Device in the Thing Registry

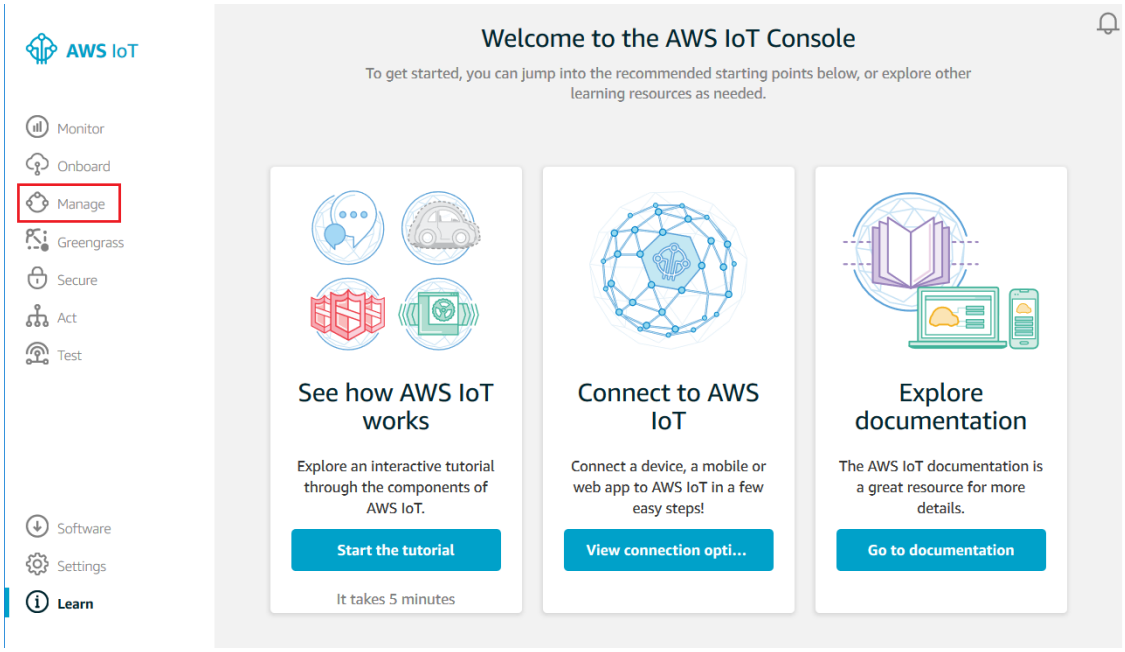
Devices connected to AWS IoT are represented by things in the thing registry. The thing registry allows you to keep a record of all of the devices that are connected to your AWS IoT account.

The fastest way to start using your AWS IoT Button is to download the mobile app for iOS or Android. The mobile app creates the required AWS IoT resources for you, and adds an event source to your button that uses a Lambda blueprint to invoke a new AWS Lambda function of your choice. Blueprints are preconfigured Lambda functions that allow you to quickly connect the click of a button to the functions that fit you best, such as sending automated emails or text messages or deploying other AWS services. You can download the mobile apps from The Apple App Store or Google Play.

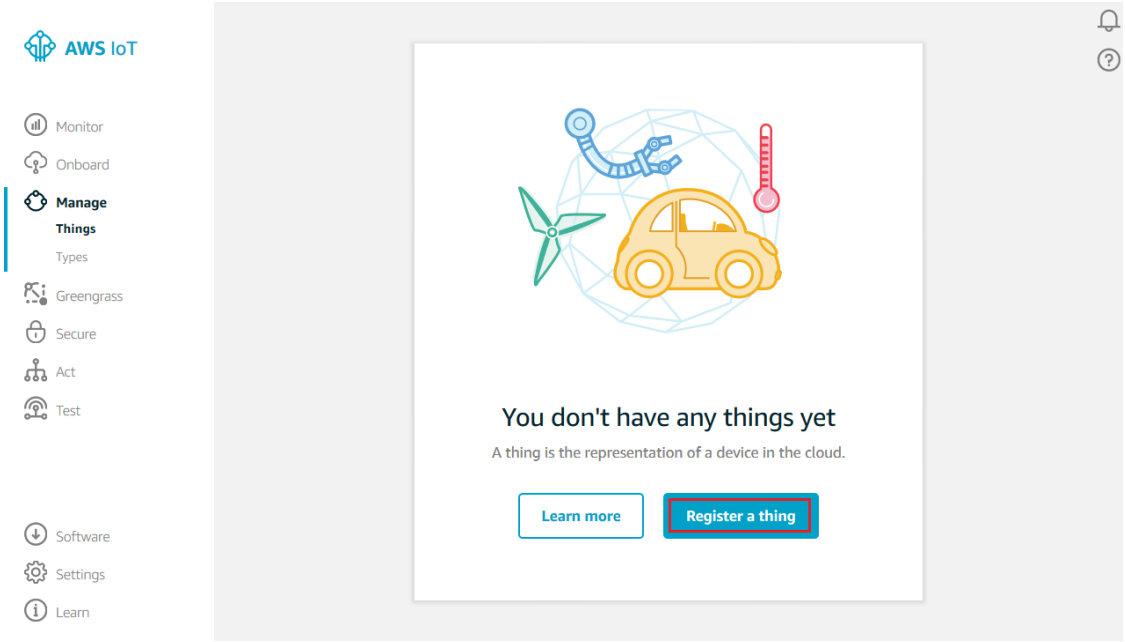
If you are unable to use the mobile apps, follow these instructions.

To register your device in the thing registry:

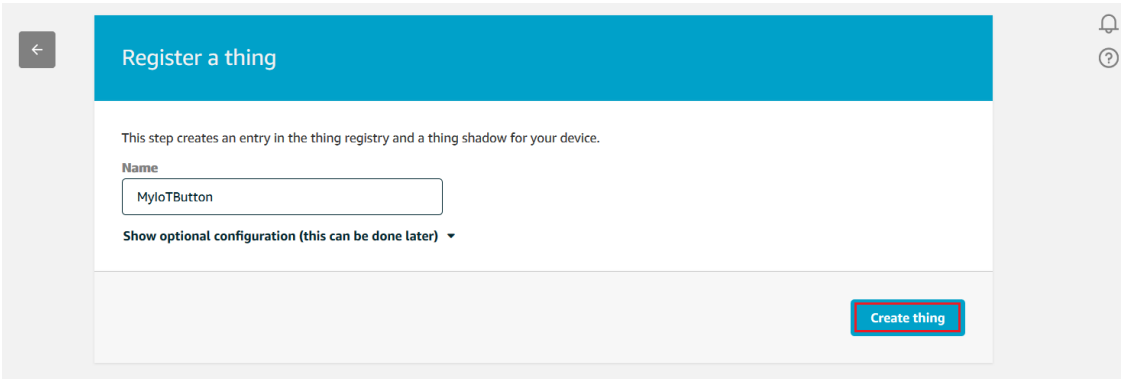
1. On the **Welcome to the AWS IoT Console** page, in the left navigation pane, choose **Registry** to expand the choices, and then choose **Things**.



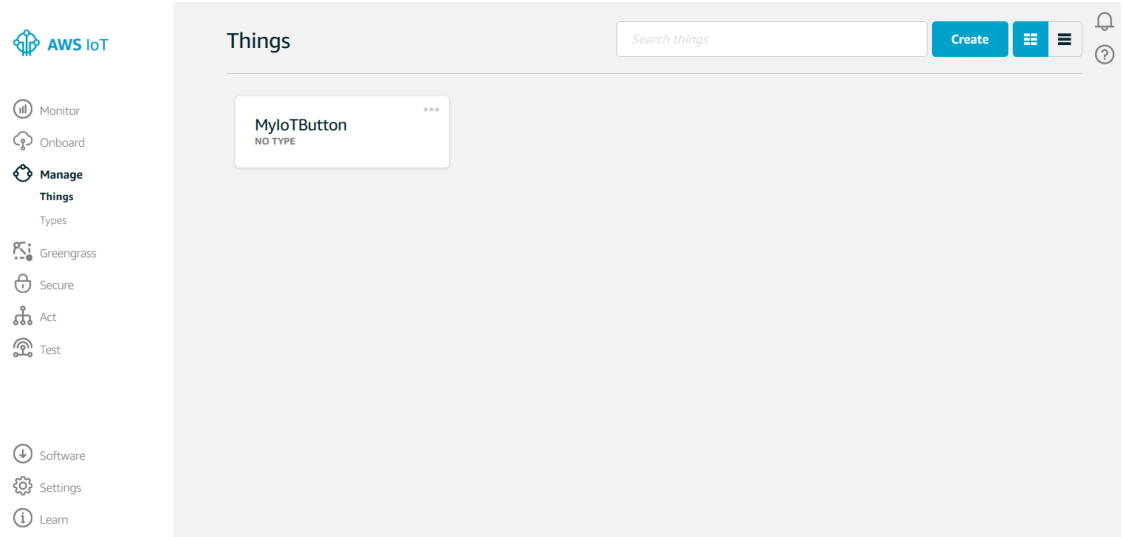
2. On the page that says **You don't have any things yet**, choose **Register a thing**.



3. On the **Register a thing** page, in the **Name** field, type a name for your device, such as **`MyIoTButton`**. Choose **Create thing** to add your device to the thing registry.



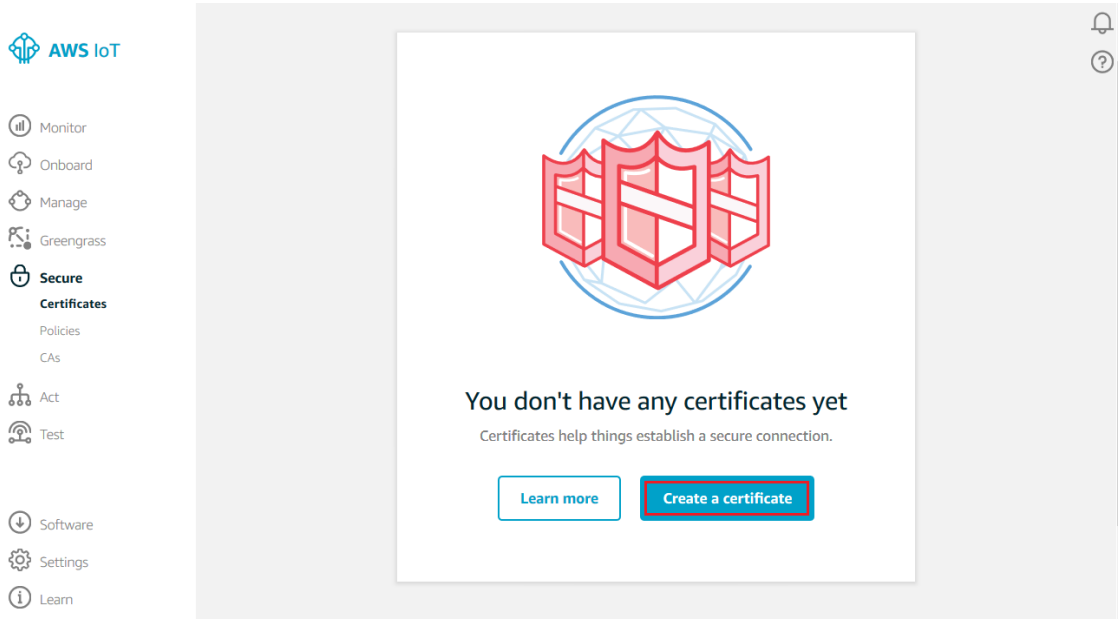
This results in the following.



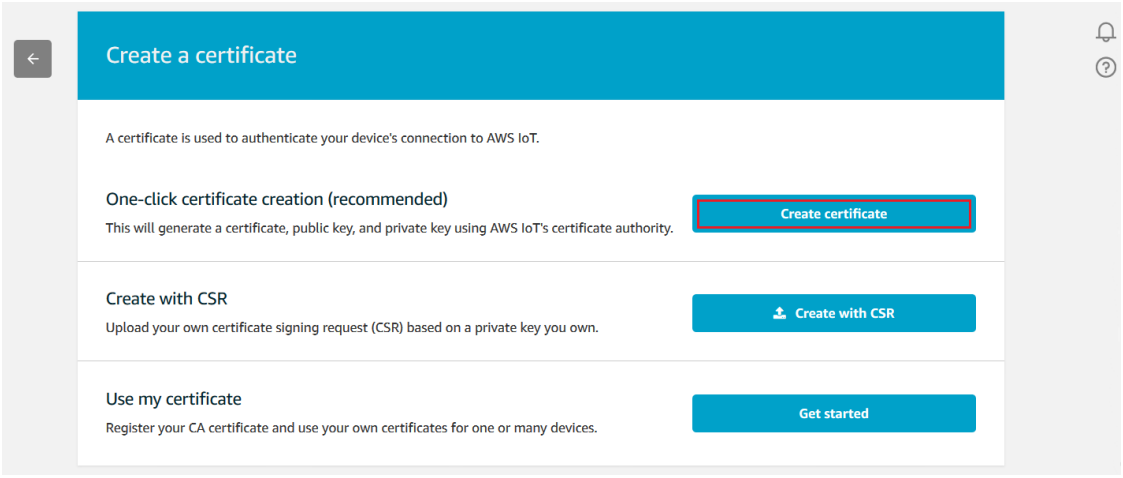
Create and Activate a Device Certificate

Communication between your device and AWS IoT is protected through the use of X.509 certificates. AWS IoT can generate a certificate for you or you can use your own X.509 certificate. In this tutorial, AWS IoT generates the X.509 certificate for you. Certificates must be activated prior to use.

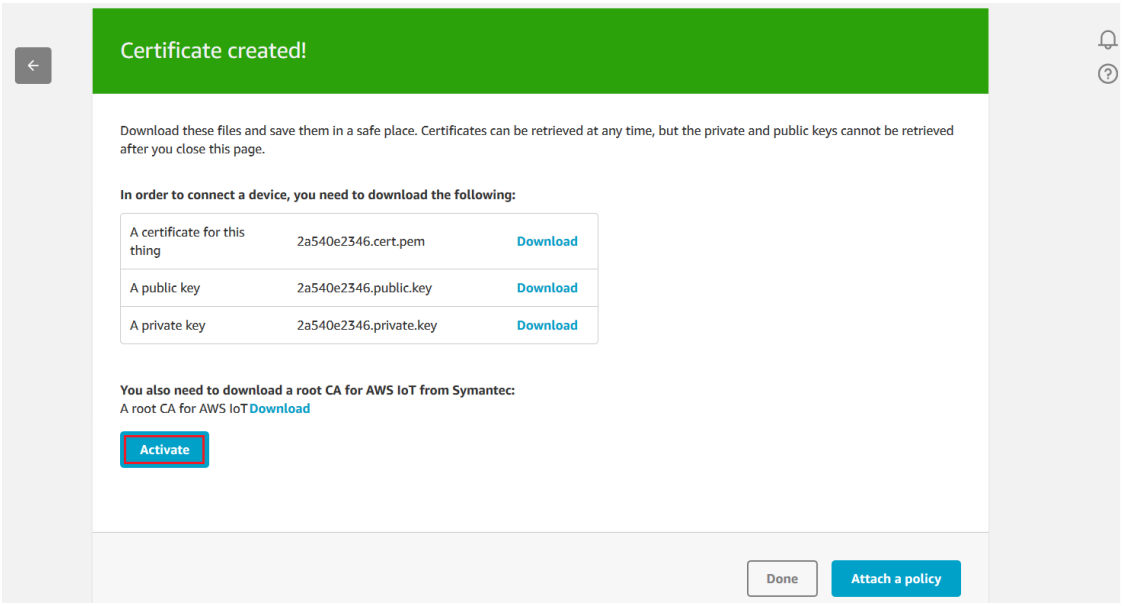
1. In the left navigation pane, choose **Secure**, **Certificates** (as necessary), and then **Create a certificate**.



2. On the **Create a certificate** page, choose **Create certificate**.



3. On the **Certificate created!** page, choose **Download** for the certificate, private key, and the root CA for AWS IoT (the public key need not be downloaded). Save each of them to your computer, and then choose **Activate** to continue.



Be aware that the downloaded filenames may appear differently then those listed on the **Certificate created!** page. For example:

- 2a540e2346-certificate.pem.crt.txt
- 2a540e2346-private.pem.key
- 2a540e2346-public.pem.key

> **Note**
> Although it is unlikely, root CA certificates are subject to expiration and/or revocation. If this should occur, you must copy new a root CA certificate onto your device.

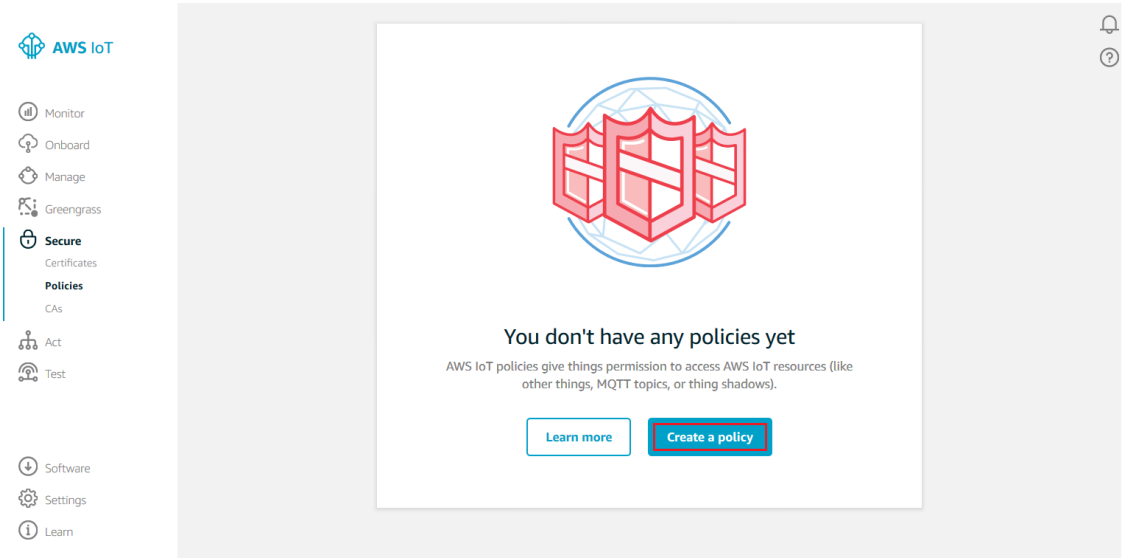
4. Choose **Done**.

Create an AWS IoT Policy

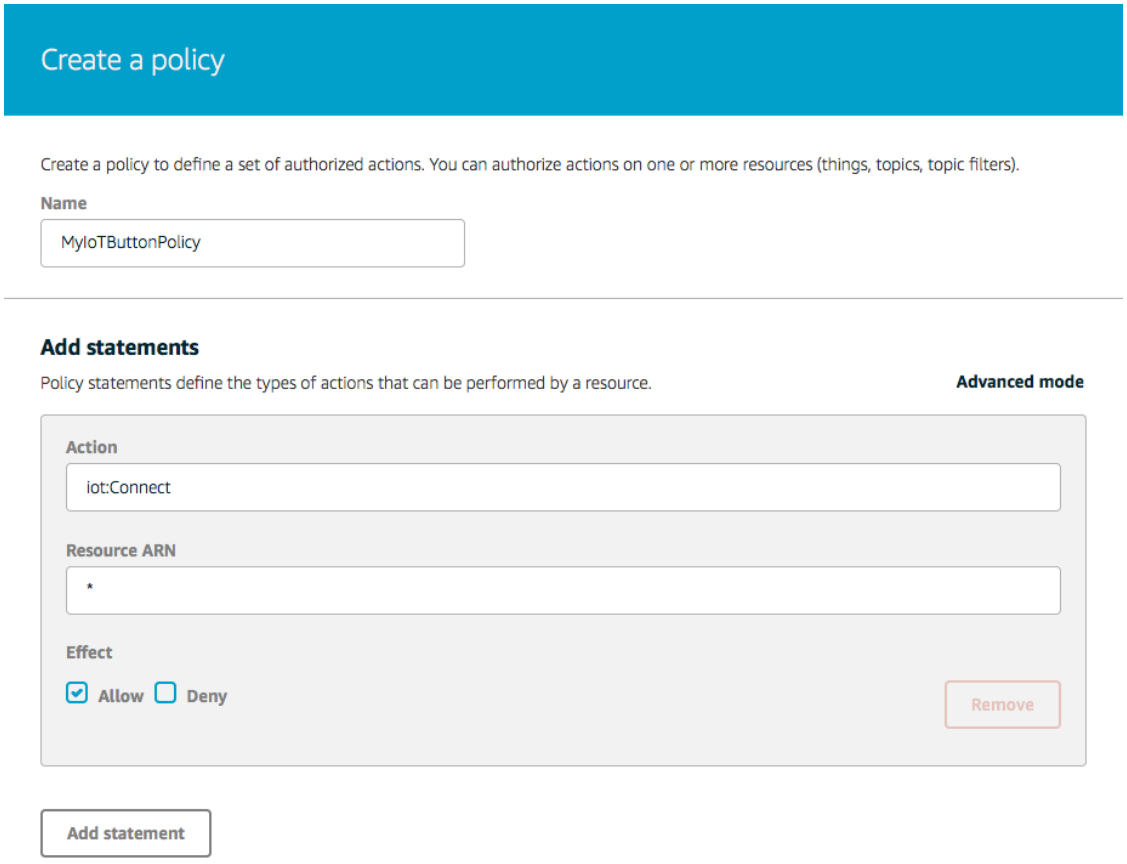
X.509 certificates are used to authenticate your device with AWS IoT. AWS IoT policies are used to authorize your device to perform AWS IoT operations, such as subscribing or publishing to MQTT topics. Your device will presents its certificate when sending messages to AWS IoT. To allow your device to perform AWS IoT operations, you must create an AWS IoT policy and attach it to your device certificate.

To create an AWS IoT policy:

1. In the left navigation pane, choose **Secure**, and then **Policies**. On the **You don't have a policy yet** page, choose **Create a policy**.



2. On the **Create a policy** page, in the **Name** field, type a name for the policy (for example, **`MyIoTButtonPolicy`**). In the **Action** field, type **iot:Connect**. In the **Resource ARN** field, type `*`. Select the Allow checkbox. This allows all clients to connect to AWS IoT.



Note
You can restrict which clients (devices) are able to connect by specifying a client ARN as the resource. The client ARNs follow this format:

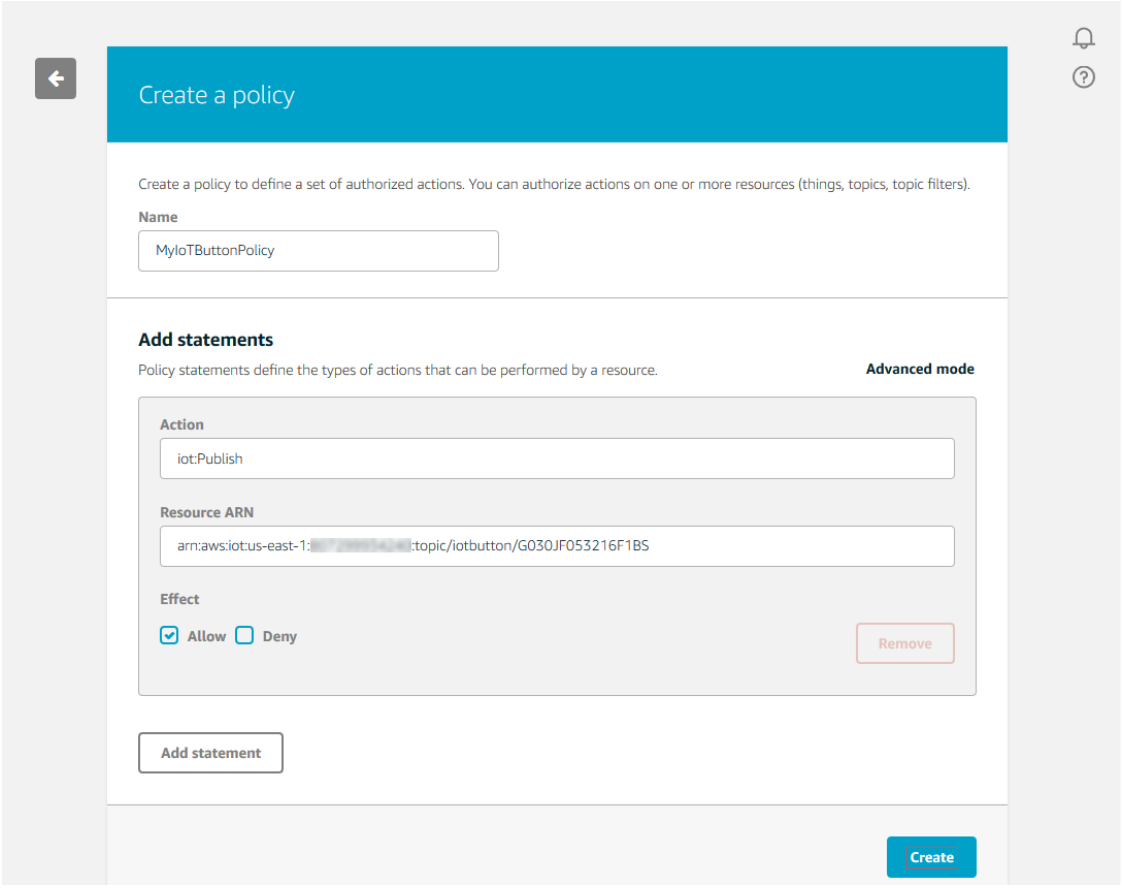
```
arn:aws:iot:your-region:your-aws-account:client/<my-client-id>
```

Select the **Add Statement** button to add another policy statement. In the **Action** field, type **iot:Publish**. In the **Resource ARN** field, type the ARN of the topic to which your device will publish.

> **Note**
> The topic ARN follows this format:
> `arn:aws:iot:your-region:your-aws-account:topic/iotbutton/your-button-serial-number`
> For example:
> `arn:aws:iot:us-east-1:123456789012:topic/iotbutton/G030JF055364XVRB`
> You can find the serial number on the bottom of your button.
> If you are not using an AWS IoT button, after `topic/` in the ARN, place the topic your device publishes to. For example:
> `arn:aws:iot:us-east-1:123456789012:topic/my/topic/here`

Finally, select the **Allow** check box. This allows your device to publish messages to the specified topic.

3. After you have entered the information for your policy, choose **Create**.

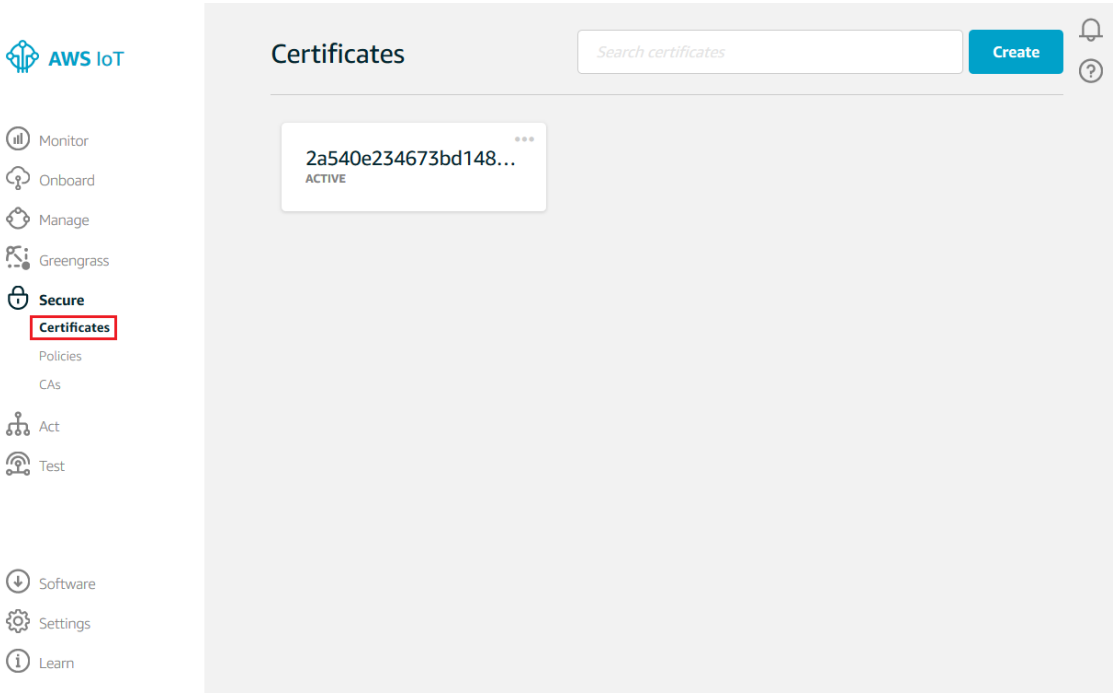


For more information, see Managing AWS IoT Policies.

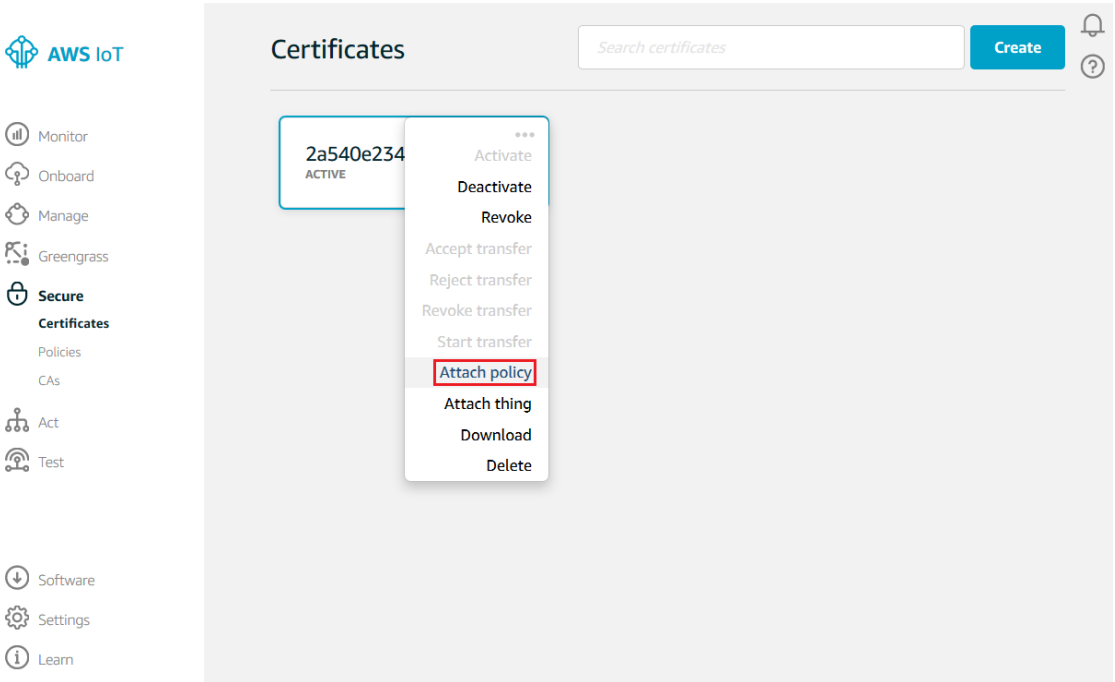
Attach an AWS IoT Policy to a Device Certificate

Now that you have created a policy, you must attach it to your device certificate. Attaching an AWS IoT policy to a certificate gives the device the permissions specified in the policy.

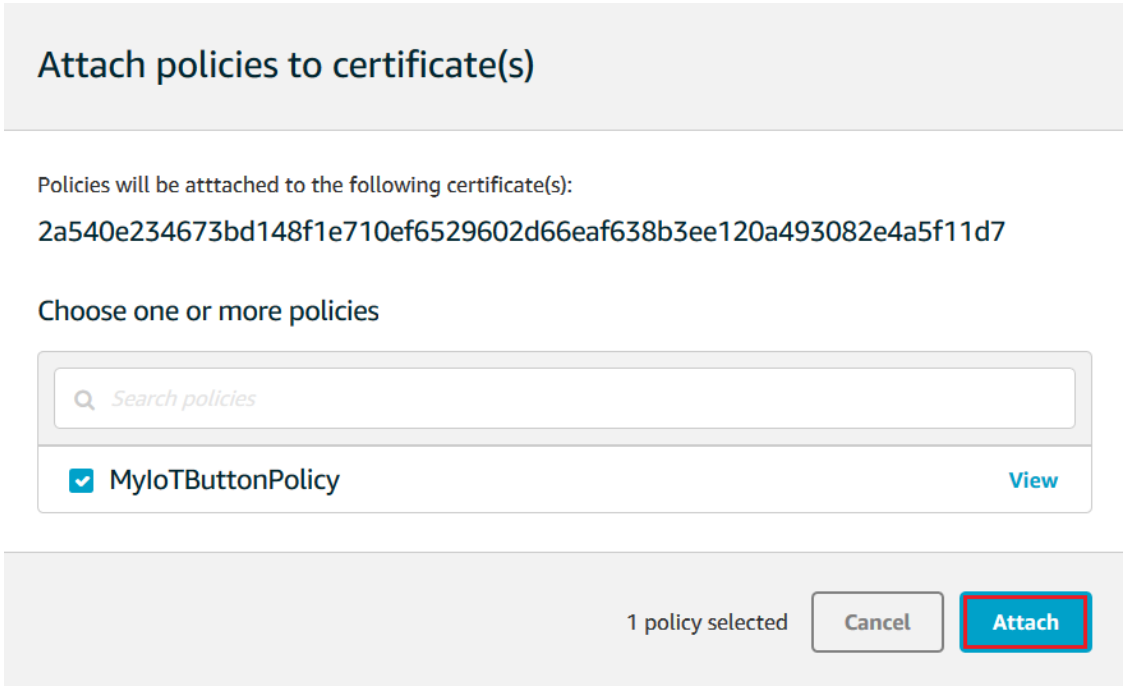
1. In the left navigation pane, choose **Secure**, and then **Certificates**.



2. In the box for the certificate you created, choose **...** to open a drop-down menu, and then choose **Attach policy**.



3. In the **Attach policies to certificate(s)** dialog box, select the check box next to the policy you created in the previous step, and then choose **Attach**.

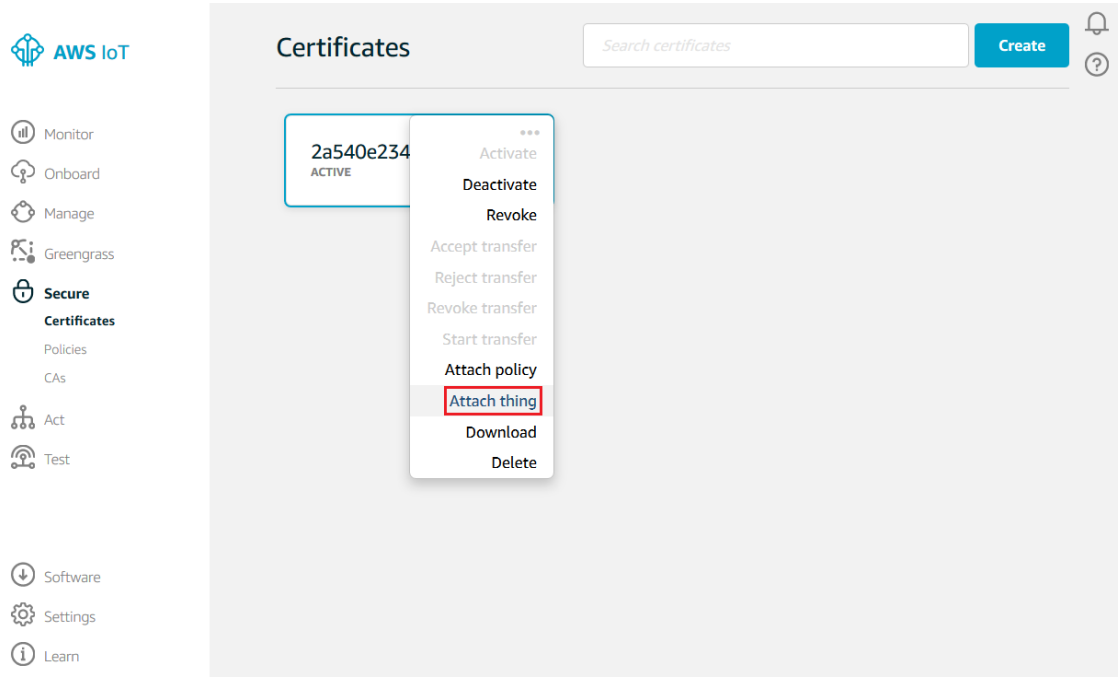


Attach a Certificate to a Thing

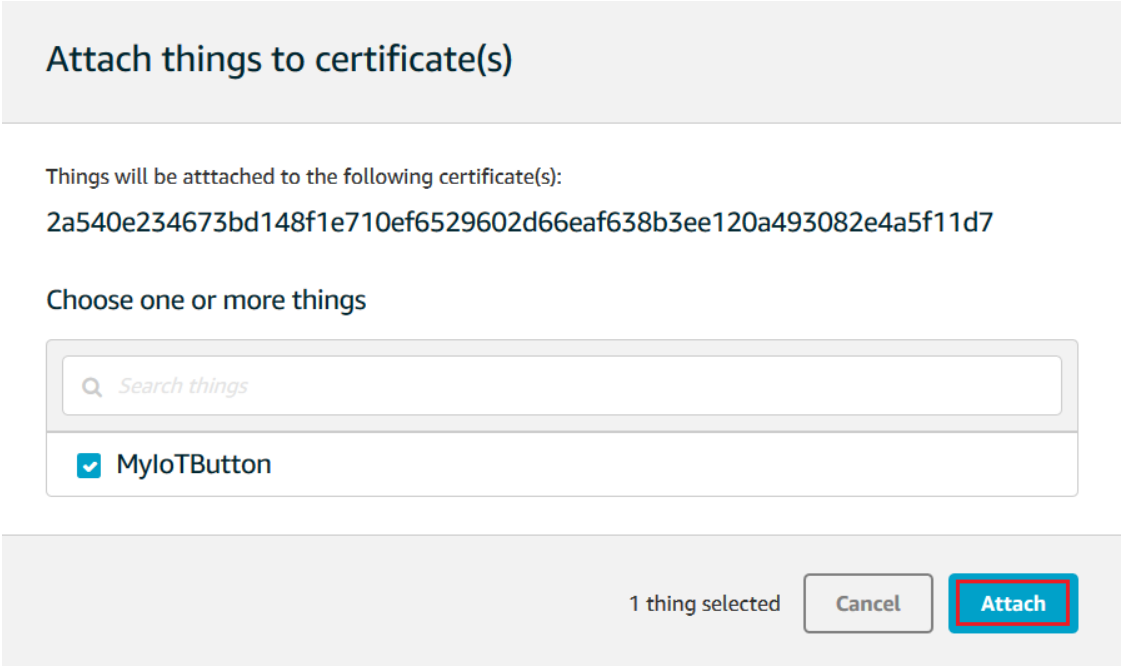
A device must have a certificate, private key and root CA certificate to authenticate with AWS IoT. We recommend that you also attach the device certificate to the thing that represents your device in AWS IoT. This allows you to create AWS IoT policies that grant permissions based on certificates attached to your things. For more information. see Thing Policy Variables (p. 127)

To attach a certificate to the thing representing your device in the thing registry:

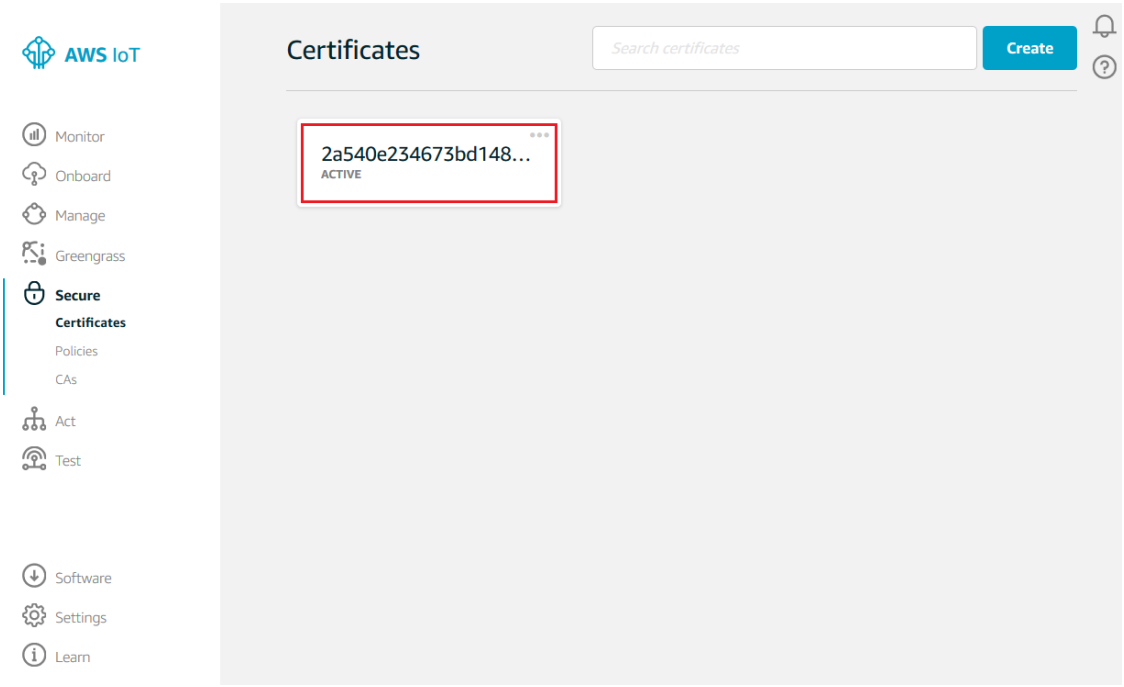
1. In the box for the certificate you created, choose **...** to open a drop-down menu, and then choose **Attach thing**.



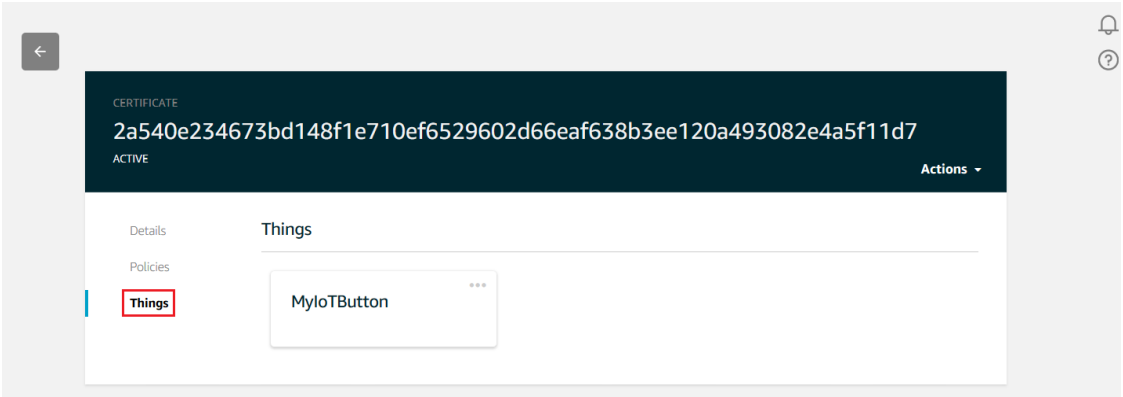
2. In the **Attach things to certificate(s)** dialog box, select the check box next to the thing you registered, and then choose **Attach**.



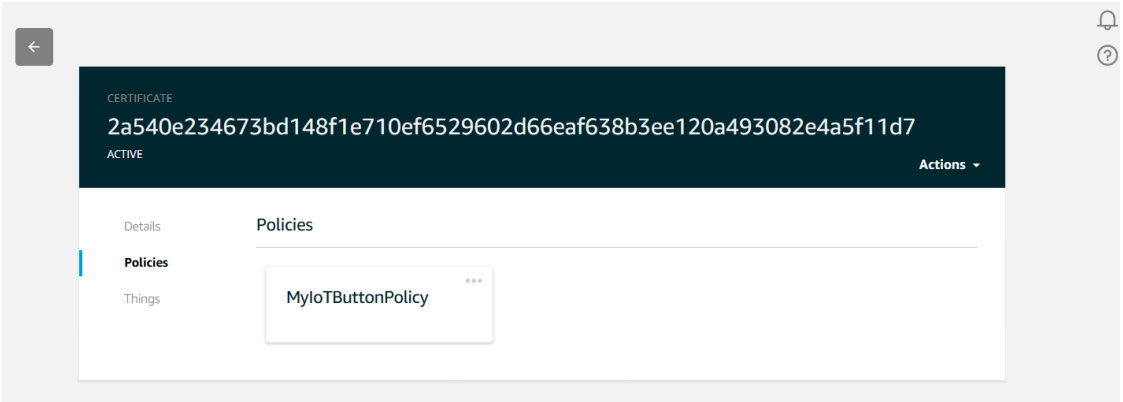
3. To verify the thing is attached, select the box representing the certificate.



4. On the **Details** page for the certificate, in the left navigation pane, choose **Things**.



5. To verify the policy is attached, on the **Details** page for the certificate, in the left navigation pane, choose **Policies**.



Configure Your Device

Configuring your device allows it to connect to your Wi-Fi network. Your device must be connected to your Wi-Fi network to install required files and send messages to AWS IoT. All devices must install a device certificate, private key, and root CA certificate in order to communicate with AWS IoT.

> **Note**
> Pressing the AWS IoT button for 15 seconds will reset it and you will have to reconfigure your Wi-Fi and device certificate.

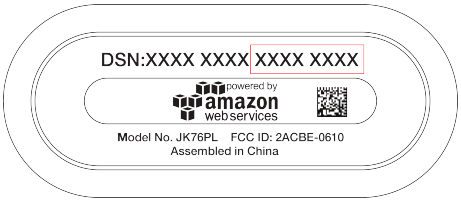
Configure an AWS IoT Button

The easiest way to configure your AWS IoT button is to use the AWS IoT button smart phone app. You can download it from the Apple App Store or the Google Play Store. If you are unable to use the smart phone app, follow these directions to configure your button.

Turn on your device

1. Remove the AWS IoT button from its packaging, and then press and hold the button until a blue blinking light appears. (This should take no longer than 15 seconds.)
2. The button acts as a Wi-Fi access point, so when your computer searches for Wi-Fi networks, it will find one called **Button ConfigureMe - XXX** where *XXX* is a three-character string generated by the button. Use your computer to connect to the button's Wi-Fi access point.

   > **Note**
   > When the blue light stops blinking, the button ceases presenting itself as a Wi-Fi access point. Therefore, if you can't complete the following procedure soon enough, you may need to invoke the blue blinking light a few times to do so. Once configured, the device does not need to present itself as a Wi-Fi access point and communicates to the internet, just like any other computer, using your local Wi-Fi network.
3. The first time you connect to the button's Wi-Fi access point, you will be prompted for the WPA2-PSK password. Type the last 8 characters of the device serial number (DSN). You'll find the DSN on the back of the device, as shown here:



Copy your device certificate and private key onto your AWS IoT button

To connect to AWS IoT, you must copy your device certificate and private key onto the AWS IoT button.

1. In a browser, navigate to http://192.168.0.1/index.html.
2. Complete the configuration form:

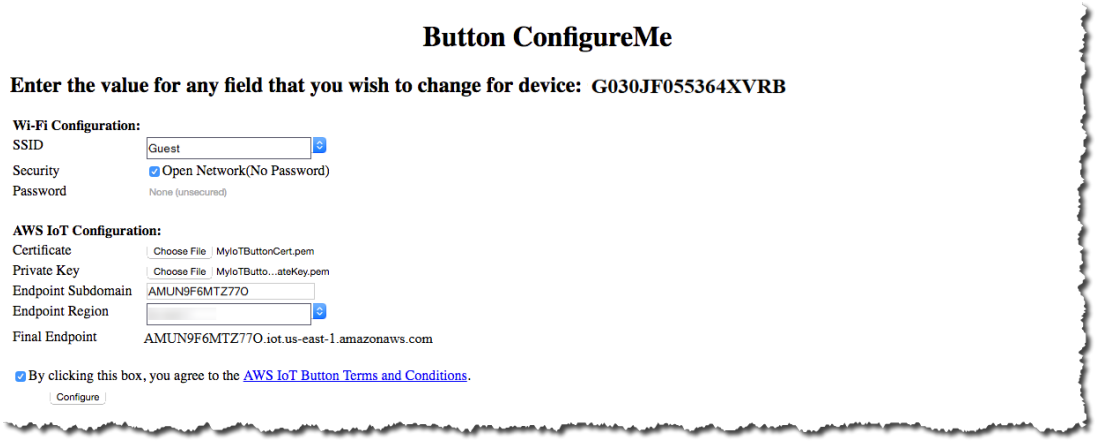
   - Type your Wi-Fi SSID and password.
   - Browse to and select your certificate and private key. For example, `2a540e2346-certificate.pem.crt.txt` and `2a540e2346-private.pem.key`, respectively.
   - Find your custom endpoint in the AWS IoT console. (From the dashboard, in the left navigation pane, choose **Manage**, and then choose **Things**. Select the box representing your button to show

its details page. On the details page, in the left navigation pane, choose **Interact** and look for the **HTTPS** section, near the top.) Your endpoint will look something like the following:

```
ABCDEFG1234567.iot.us-east-2.amazonaws.com
```

where `ABCDEFG1234567` is the subdomain and `us-east-2` is the region.

- On the **Button ConfigureMe** page, type the subdomain, and then choose the region that matches the region in your AWS IoT endpoint.
- Select the **Terms and Conditions** check box. Your settings should now look like the following:



- Choose **Configure**. Your button should now connect to your Wi-Fi network.

Configure a Different Device

Consult your device's documentation to connect to it and copy your device certificate, private key, and root CA certificate onto your device.

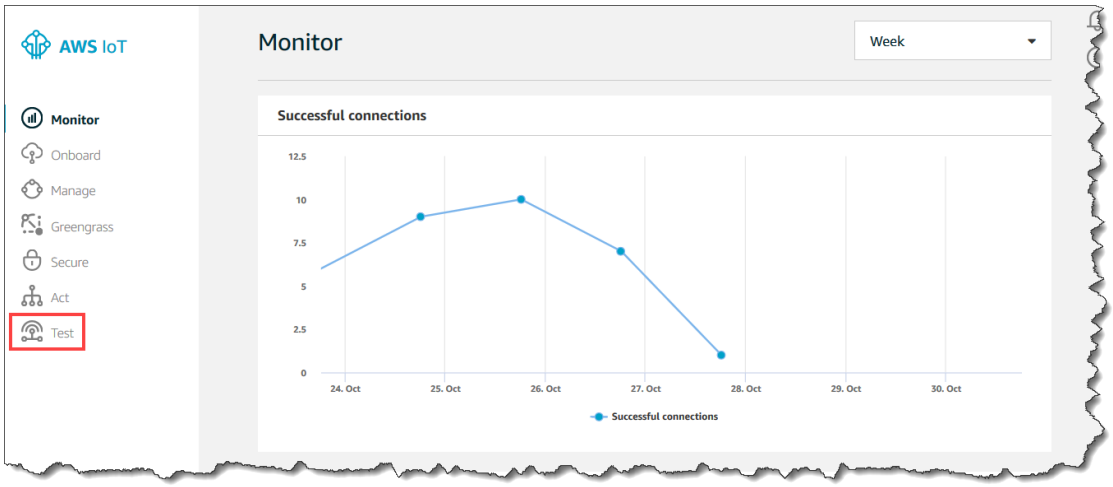
View Device MQTT Messages with the AWS IoT MQTT Client

You can use the AWS IoT MQTT client to better understand the MQTT messages sent by a device.

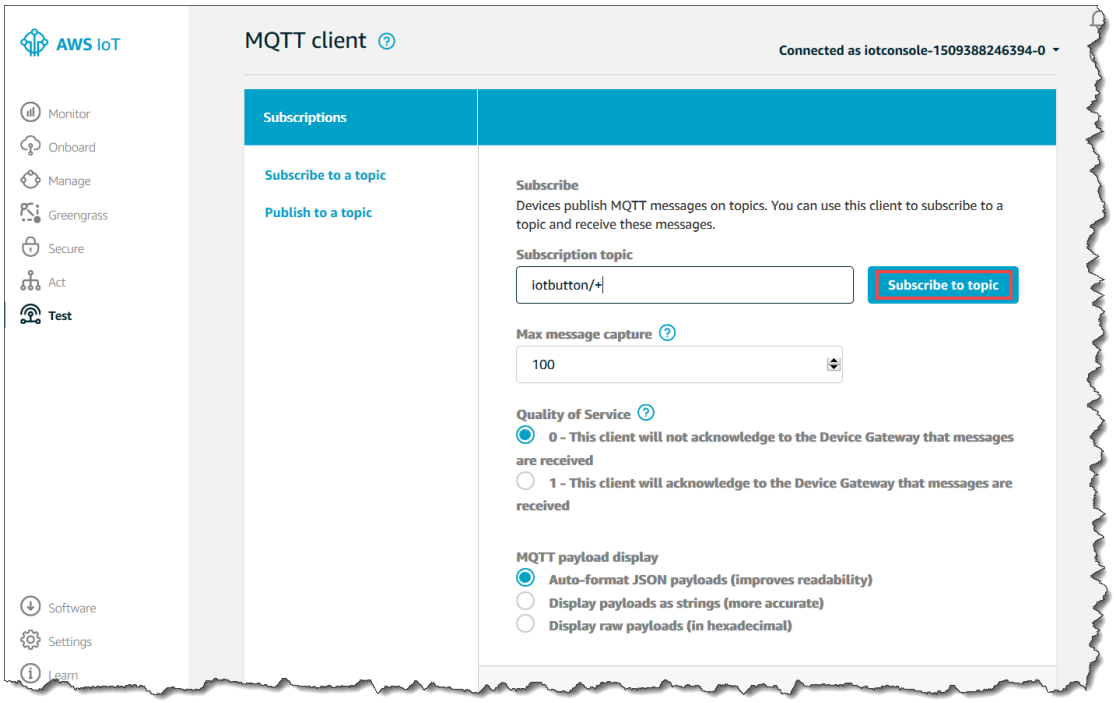
Devices publish MQTT messages on topics. You can use the AWS IoT MQTT client to subscribe to these topics to see these messages.

To view MQTT messages:

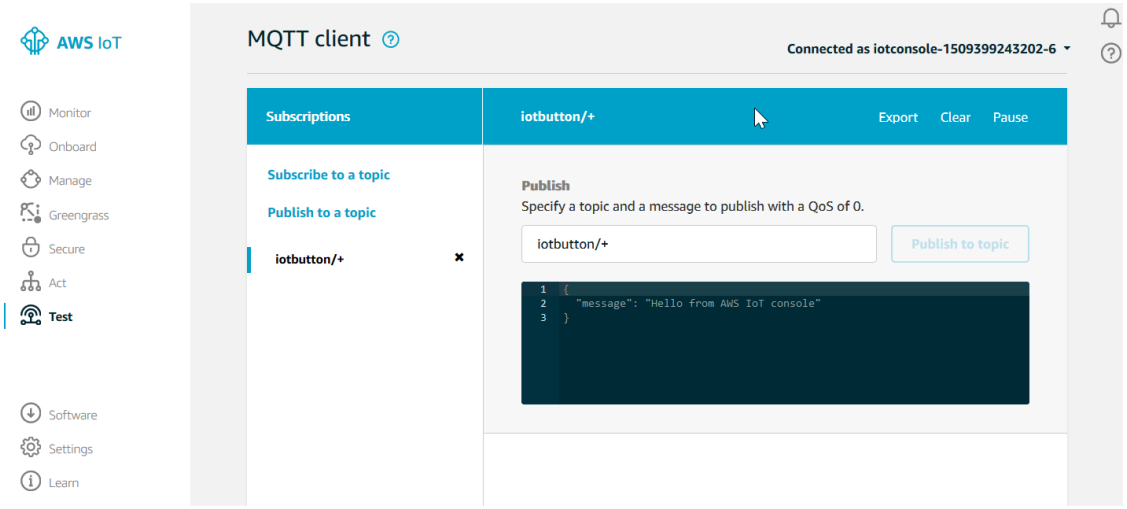
1. In the AWS IoT console, in the left navigation pane, choose **Test**.



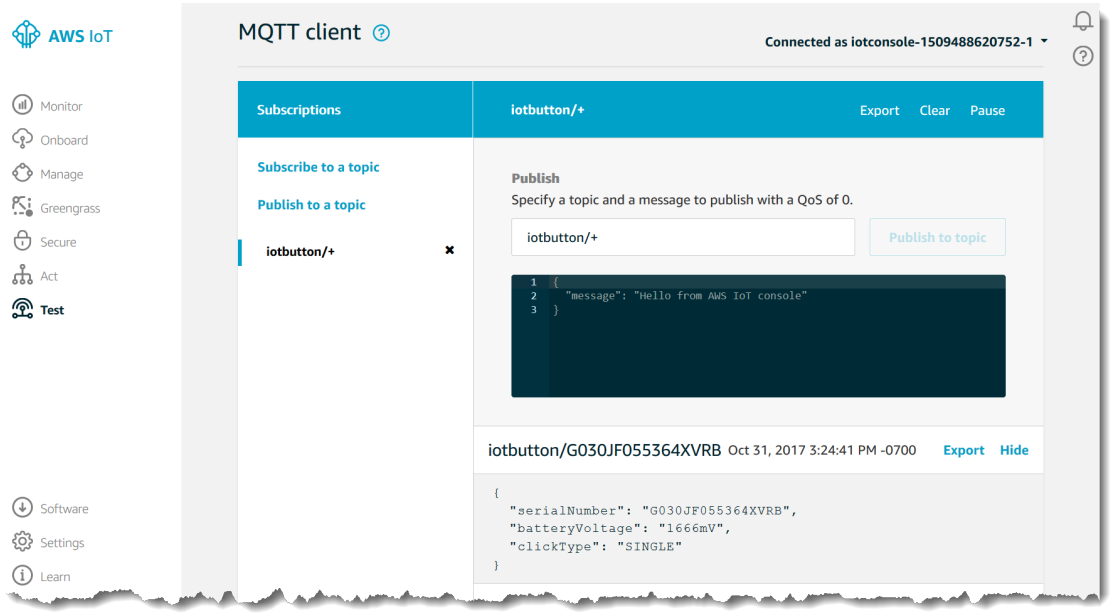
2. Subscribe to the topic on which your thing publishes. In the case of the AWS IoT button, you can subscribe to **`iotbutton/+`** (note that **+** is the wildcard character). In **Subscribe to a topic**, in the **Subscription topic** field, type **`iotbutton/+`**, and then choose **Subscribe to topic**.



Choosing **Subscribe to topic** above, results in the topic **`iotbutton/+`** appearing in the **Subscriptions** column.



3. Press your AWS IoT button, and then view the resulting message in the AWS IoT MQTT client. If you do not have a button, you will simulate a button press in the next step.



Note
The AWS IoT Button FAQs contains useful button LED color pattern information.

4. To use the AWS IoT console to publish a message:

   On the MQTT client page, in the **Publish** section, in the **Specify a topic and a message to publish**... field, type **`iotbutton/ABCDEFG12345`**. In the message payload section, type the following JSON:

```
{
    "serialNumber": "ABCDEFG12345",
    "clickType": "SINGLE",
    "batteryVoltage": "2000 mV"
}
```

   Choose **Publish to topic**. You should see the message in the AWS IoT MQTT client (choose **iotbutton/+** in the **Subscription** column to see the message).

Configure and Test Rules

The AWS IoT rules engine listens for incoming MQTT messages that match a rule. When a matching message is received, the rule takes some action with the data in the MQTT message (for example, writing data to an Amazon S3 bucket, invoking a Lambda function, or sending a message to an Amazon SNS topic). In this step, you will create and configure a rule to send the data received from a device to an Amazon SNS topic. Specifically, you will:

- Create an Amazon SNS topic.
- Subscribe to the Amazon SNS topic using a cell phone number.
- Create a rule that will send a message to the Amazon SNS topic when a message is received from your device.
- Test the rule using your AWS IoT button or an MQTT client.

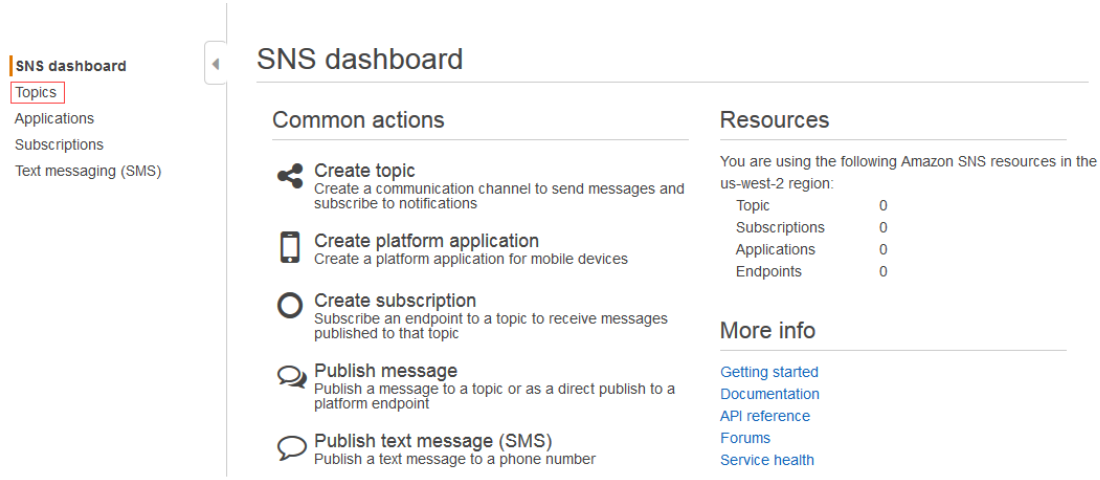
In the upper-right corner of this page, there is a **Filter View** drop-down list. For instructions for testing your rule by using the AWS IoT button, choose **AWS IoT Button**. For instructions for testing your rule by using the AWS IoT MQTT client, choose **MQTT Client**.

Create an SNS Topic

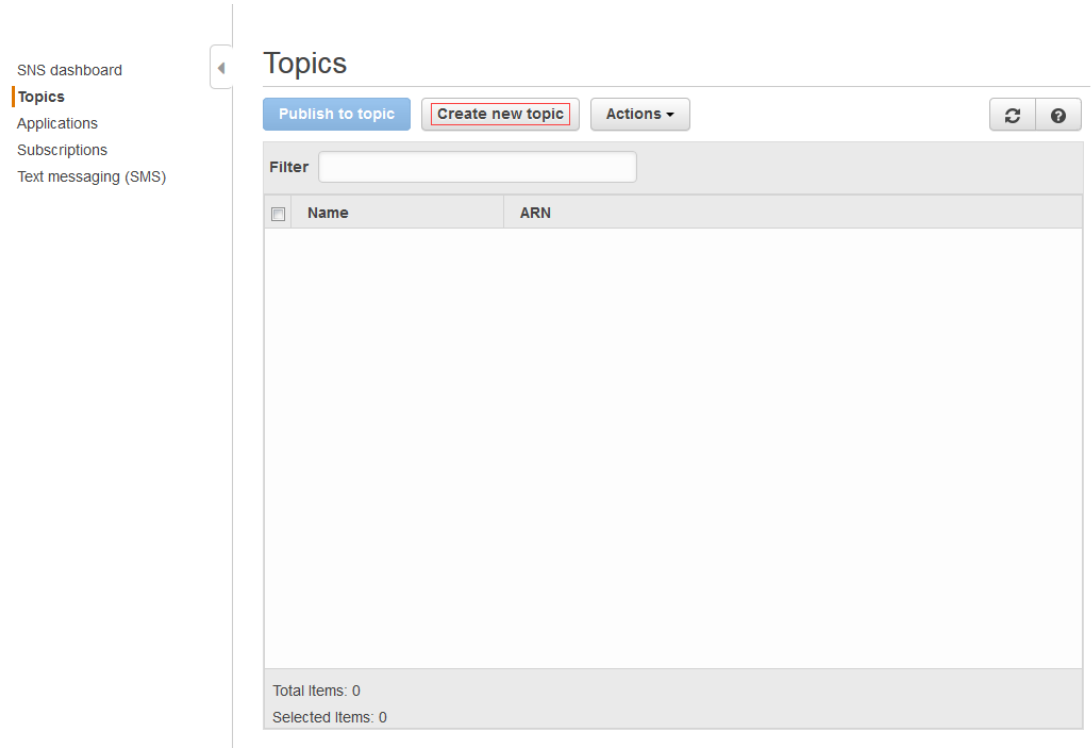
Use the Amazon SNS console to create an Amazon SNS topic.

> **Note**
> Amazon SNS is not available in all AWS regions.

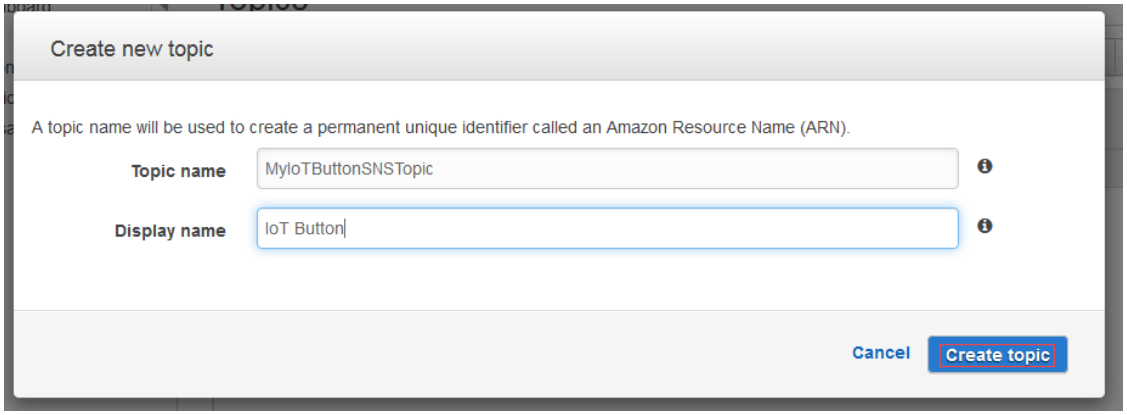
1. Open the Amazon SNS console.
2. On the left pane, choose **Topics**.



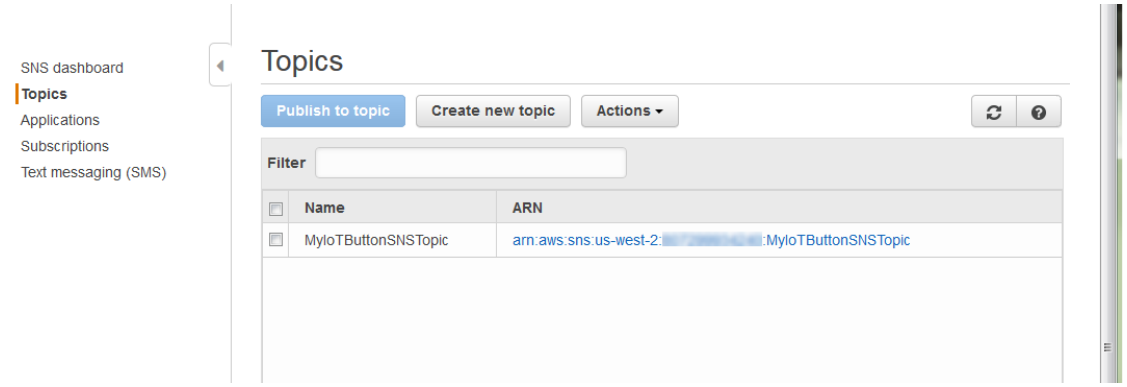
3. Choose **Create new topic**.



4. Type a topic name and a display name, and then choose **Create topic**.



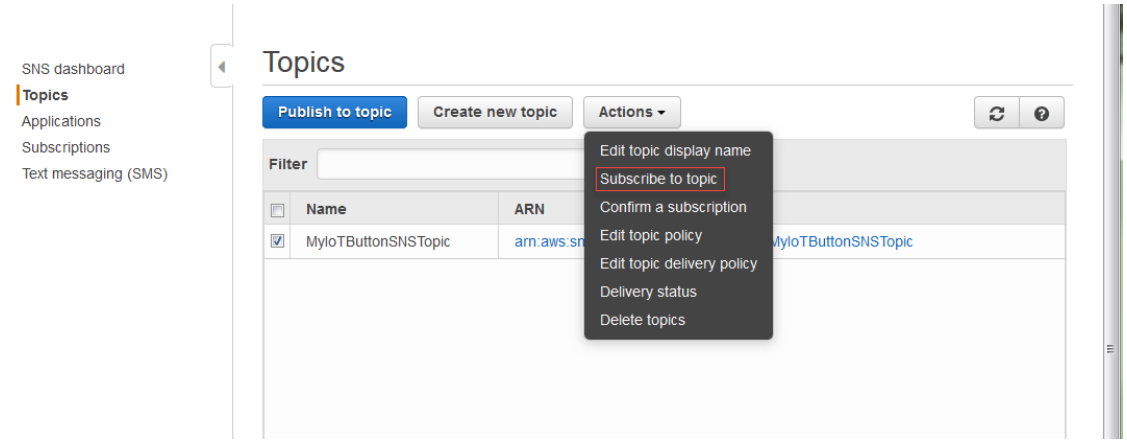
5. Make a note of the ARN for the topic you just created.



Subscribe to an Amazon SNS Topic

To receive SMS messages on your cell phone, subscribe to the Amazon SNS topic.

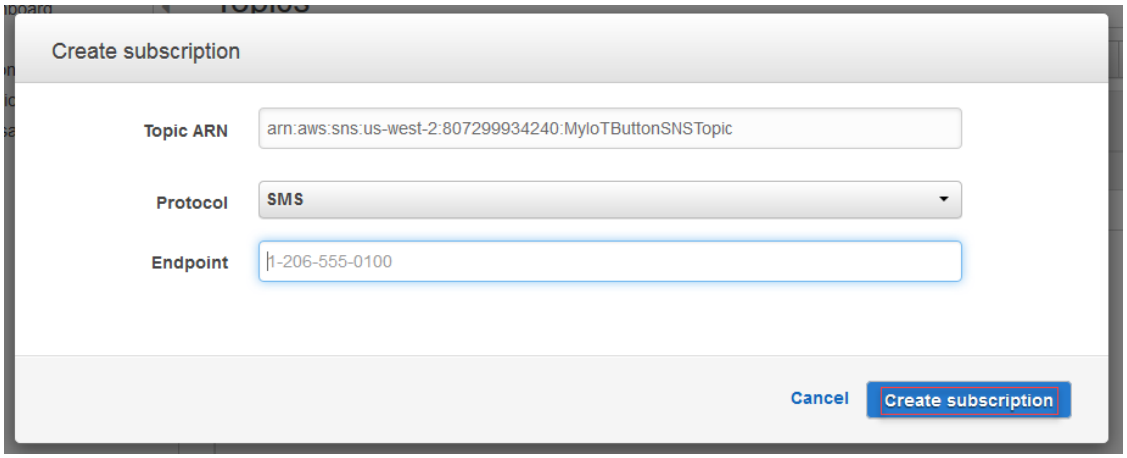
1. In the Amazon SNS console, select the check box next to the topic you just created. From the **Actions** menu, choose **Subscribe to topic**.



2. On **Create subscription**, from the **Protocol** drop-down list, choose **SMS**.

   In the **Endpoint** field, type the phone number of an SMS-enabled cell phone, and then choose **Create subscription**.

Note
Enter the phone number using numbers and dashes only.

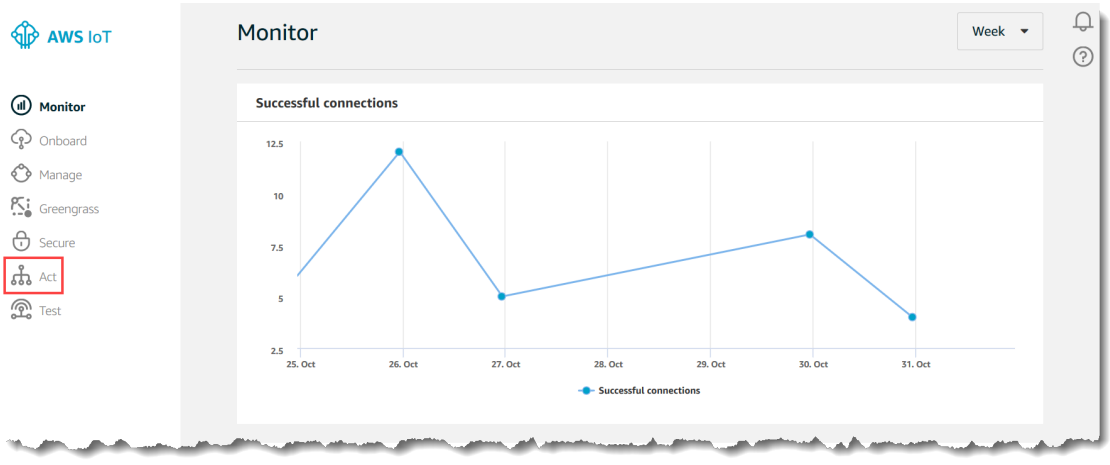


Create a Rule

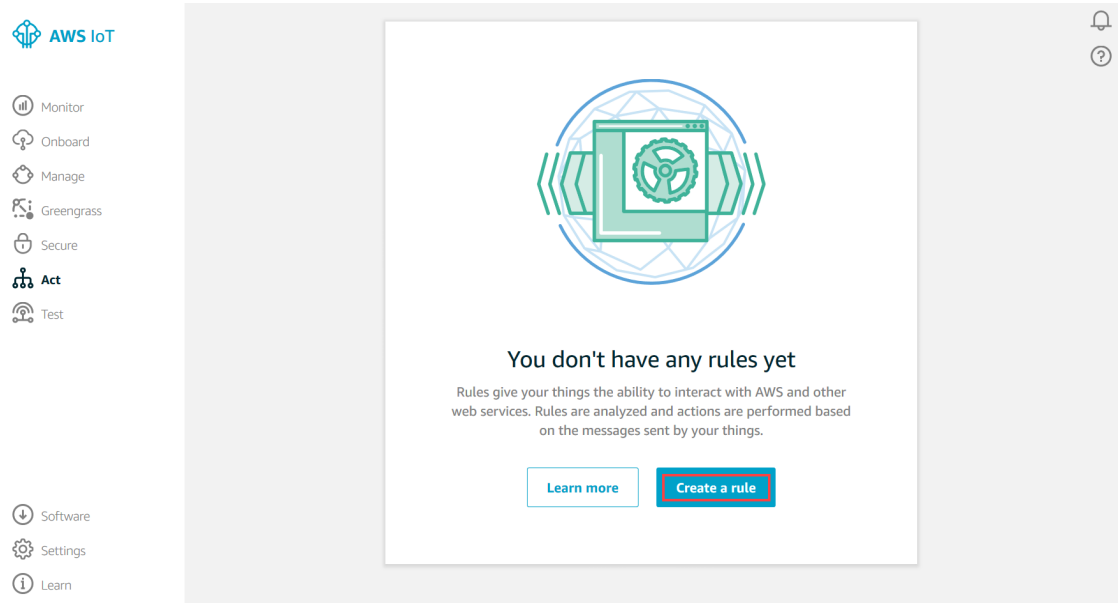
AWS IoT rules consist of a topic filter, a rule action, and, in most cases, an IAM role. Messages published on topics that match the topic filter trigger the rule. The rule action defines which action to take when the rule is triggered. The IAM role contains one or more IAM policies that determine which AWS services the rule can access. You can create multiple rules that listen on a single topic. Likewise, you can create a single rule that is triggered by multiple topics. The AWS IoT rules engine continuously processes messages published on topics that match the topic filters defined in the rules.

In this example, you will create a rule that uses Amazon SNS to send an SMS notification to a cell phone number.

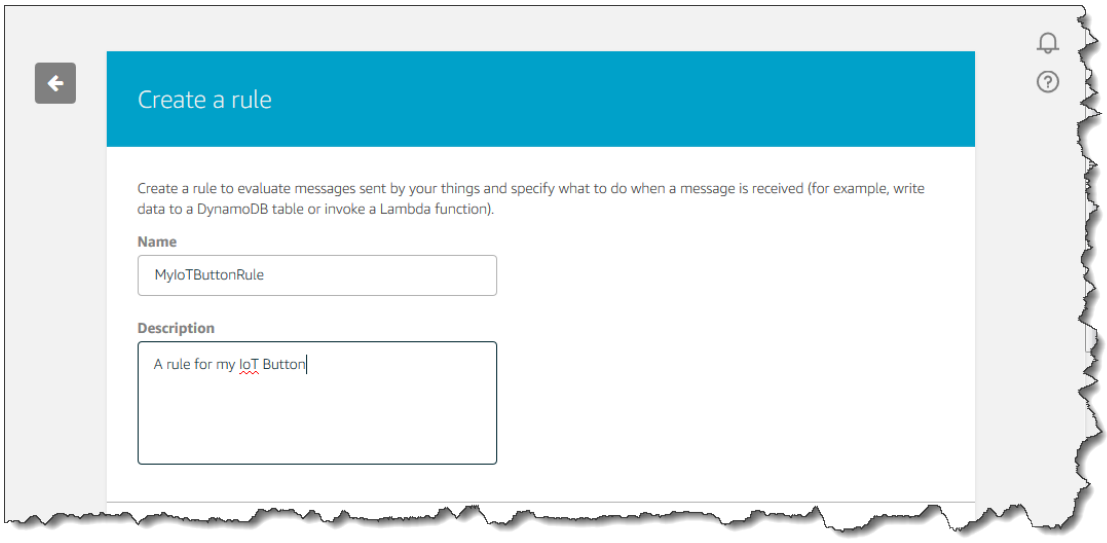
1. In the AWS IoT console, in the left navigation pane, choose **Act**.



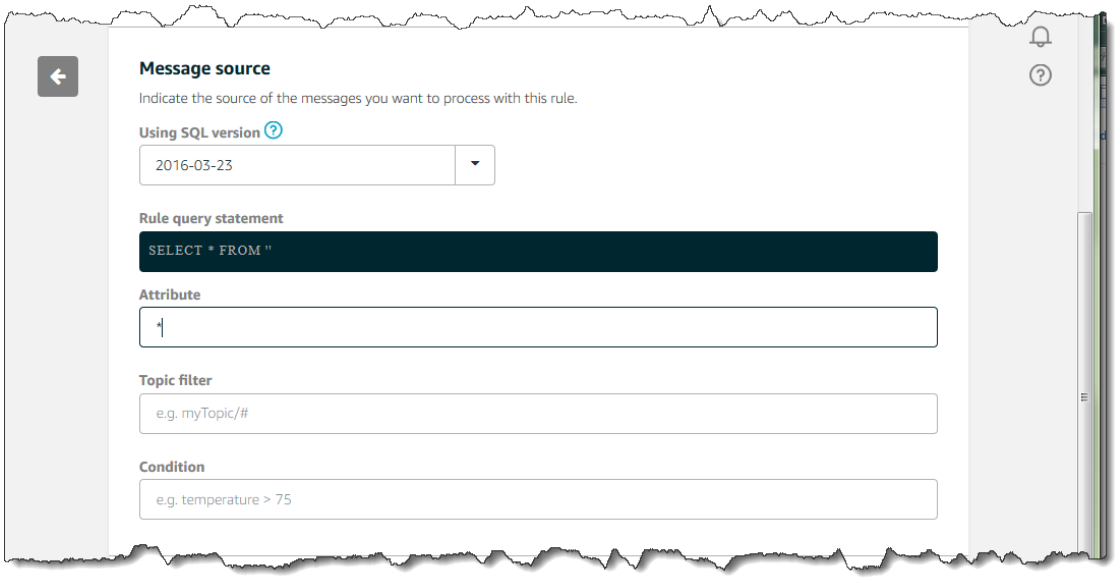
2. On the **Act** page, choose **Create a rule**.



3. On the **Create a rule** page, in the **Name** field, type a name for your rule. In the **Description** field, type a description for the rule.



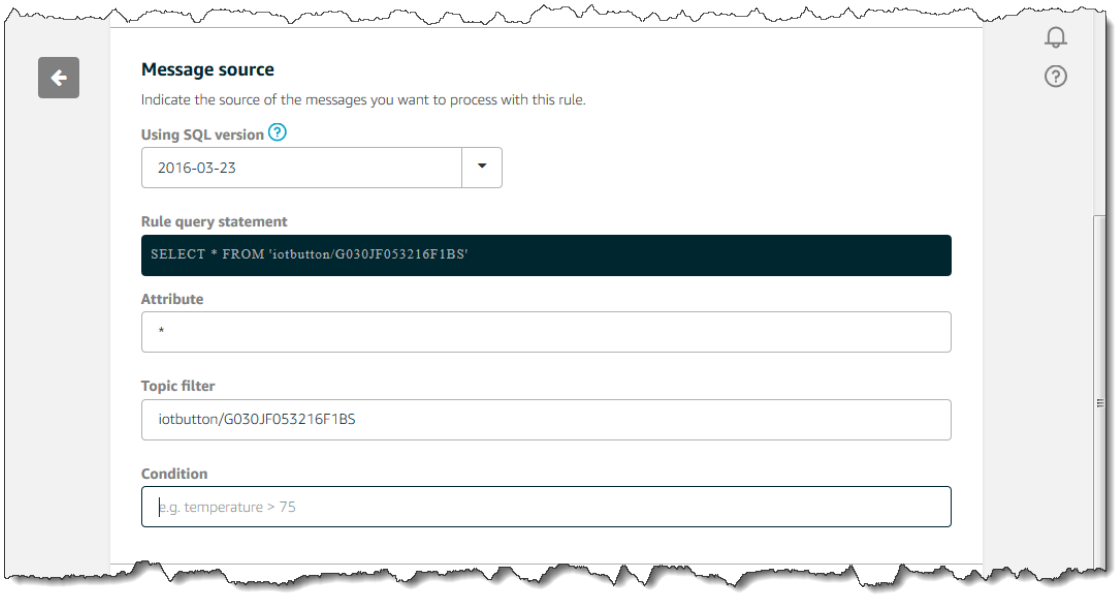
4. Scroll down to **Message source**. Choose the latest version from the **Using SQL version** drop-down list. In the **Attribute** field, type *****. This specifies that you want to send the entire MQTT message that triggered the rule.



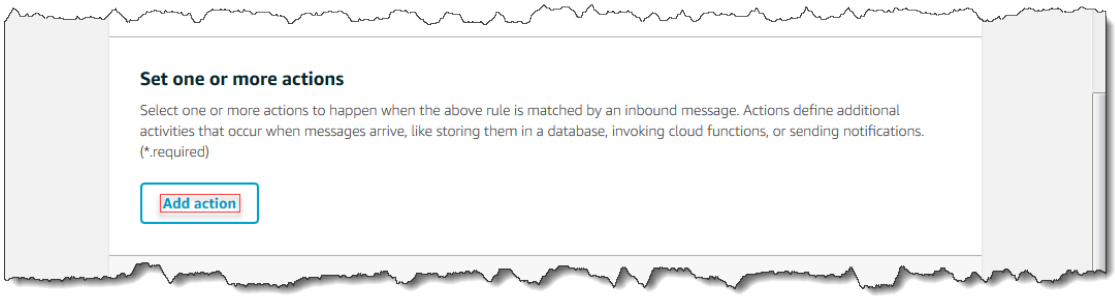
5. The rules engine uses the topic filter to determine which rules to trigger when an MQTT message is received. In the **Topic filter** field, type **`iotbutton/`*`your-button-DSN`***. If you are not using an AWS IoT button, type **`my/topic`** or the topic used in the rule.

   Note
   You can find the DSN on the bottom of the button.

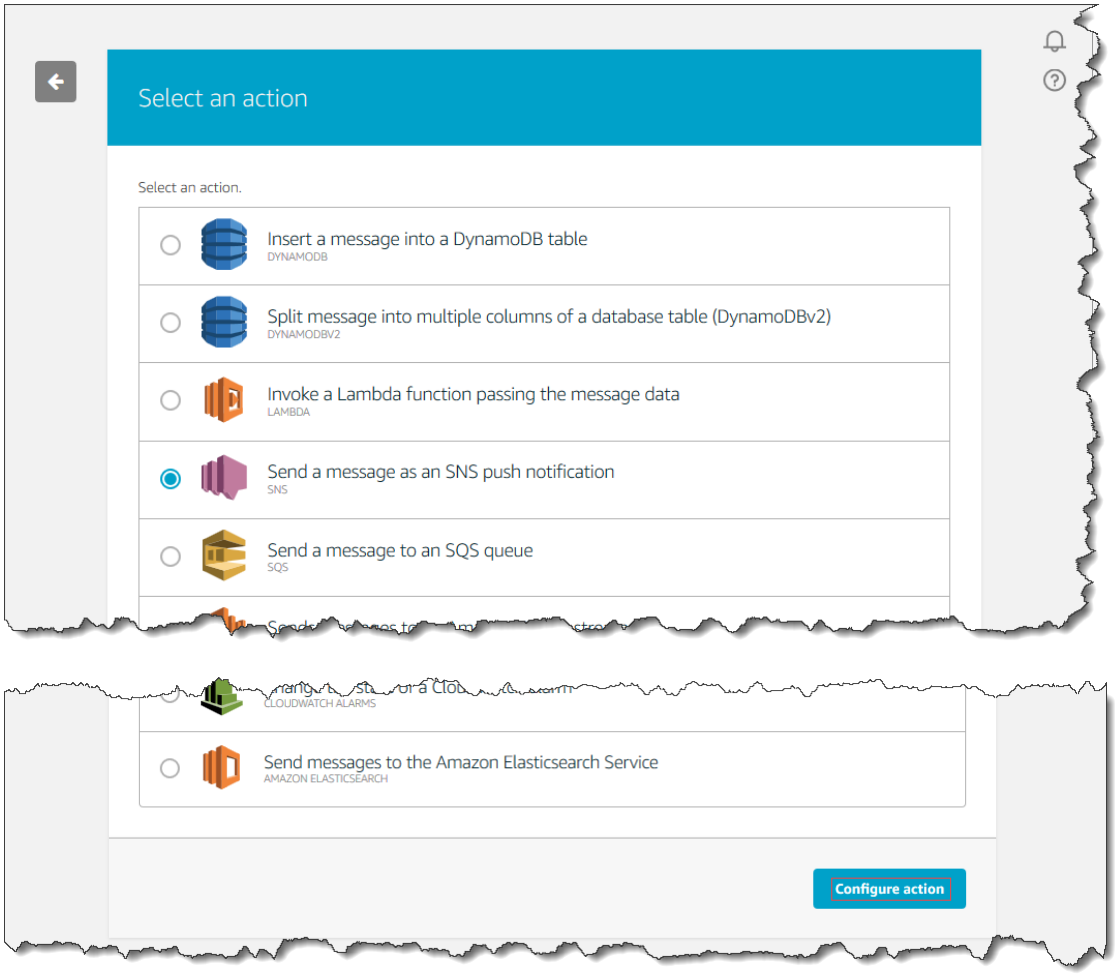
   Leave **Condition** blank.



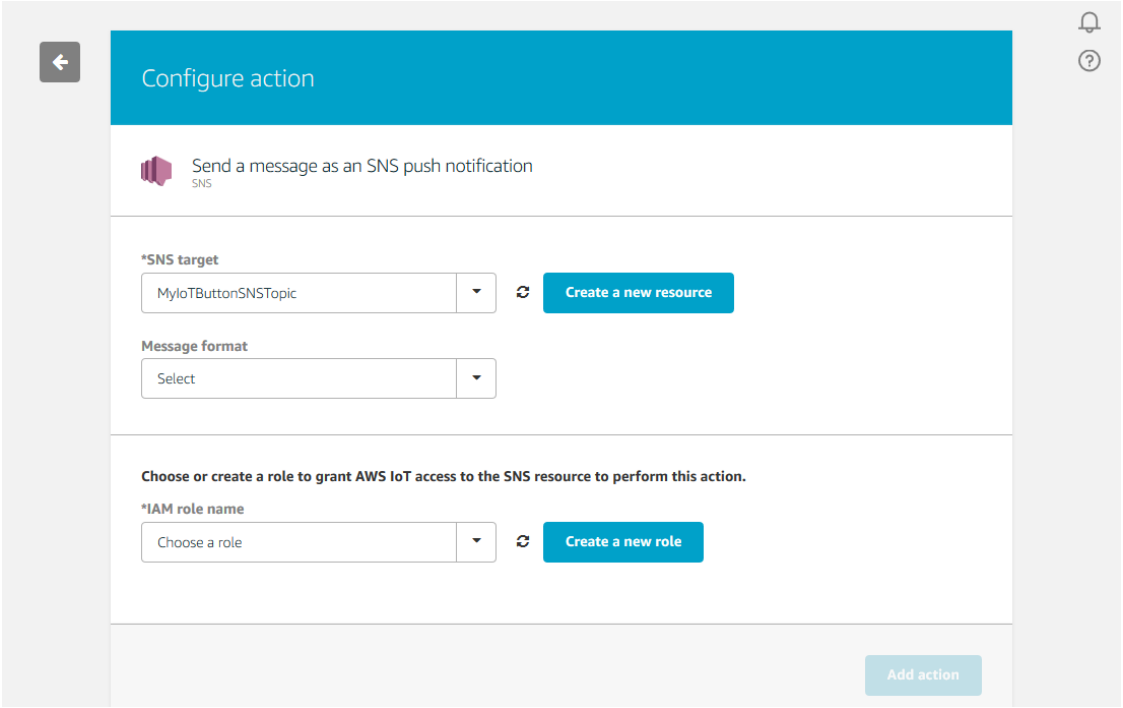
6. In **Set one or more actions**, choose **Add action**.



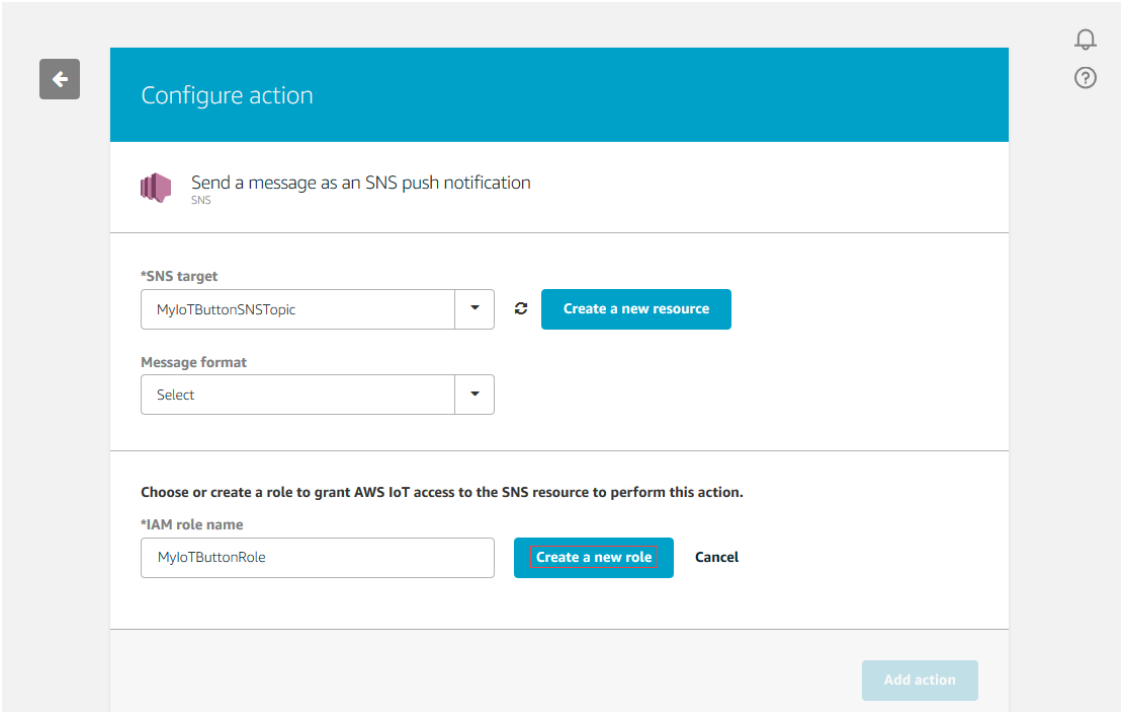
7. On the **Select an action** page, select **Send a message as an SNS push notification**, and then choose **Configure action**.



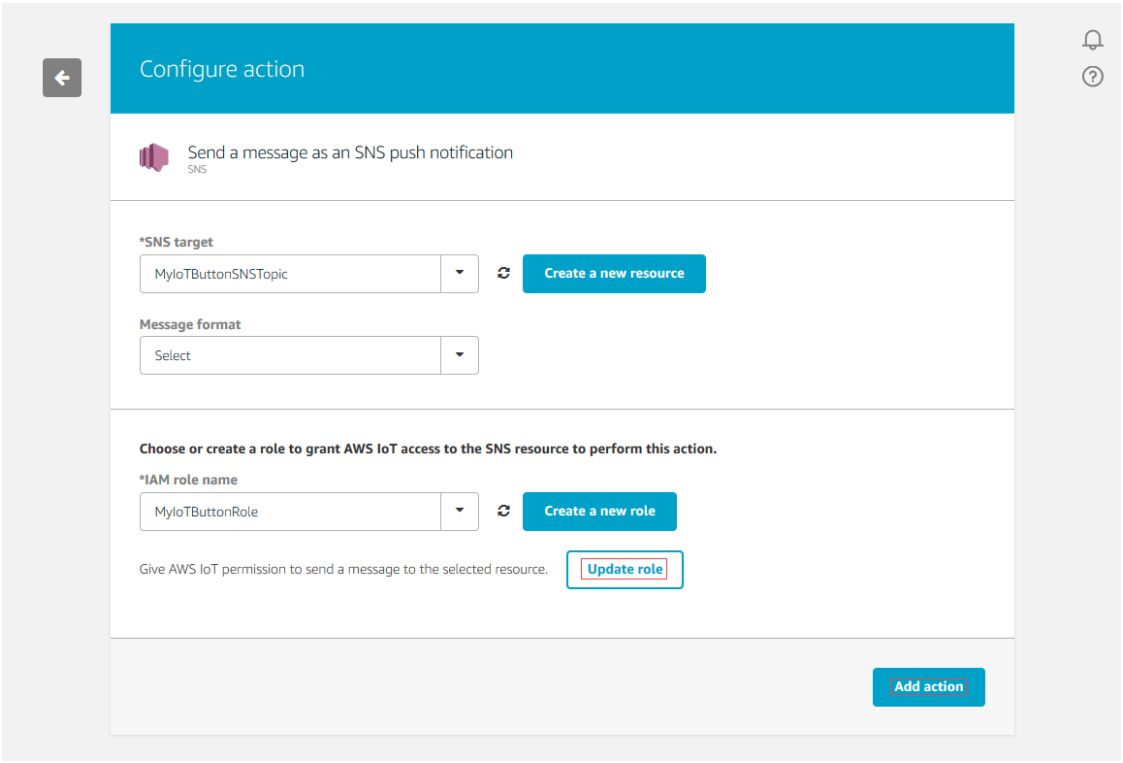
8. On the **Configure action** page, from the **SNS target** drop-down list, choose the Amazon SNS topic you created earlier.



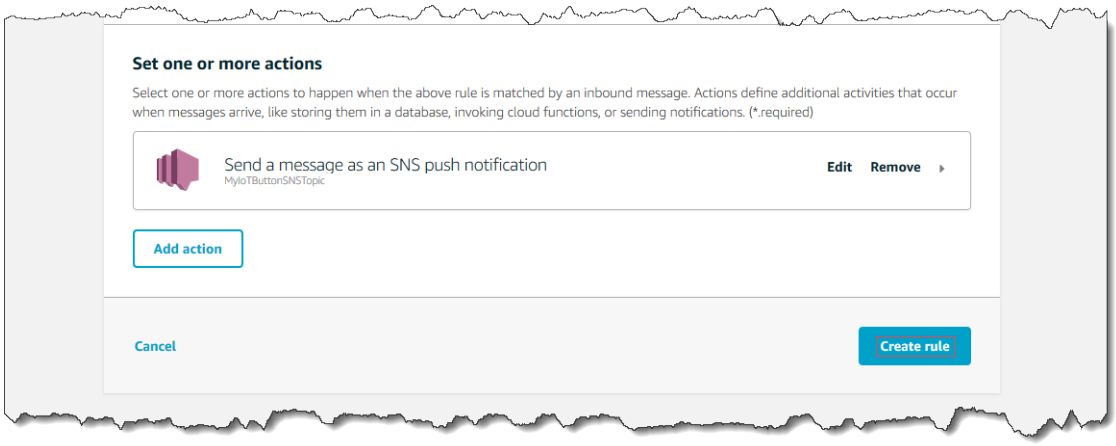
9. Now give AWS IoT permission to publish to the Amazon SNS topic on your behalf when the rule is triggered. Choose **Create a new role**. Enter a name for your new role in the **IAM role name** field. After you have entered the name, choose **Create a new role** again. Select the newly created role from the **IAM role name** drop-down list.



10. Choose **Update role** to apply the permissions to the newly created role, and then choose **Add action**.



11. On the **Create a Rule** page, choose **Create rule**.



For more information about creating rules, see AWS IoT Rules .

Test the Amazon SNS Rule

You can test your rule by using an AWS IoT button or the AWS IoT MQTT client.

AWS IoT Button

Press your button. You should receive an SMS text that shows the current battery charge level on your device (among other things). Try a long press (about 2 seconds) and a fast double pressing, and note the resulting messages.

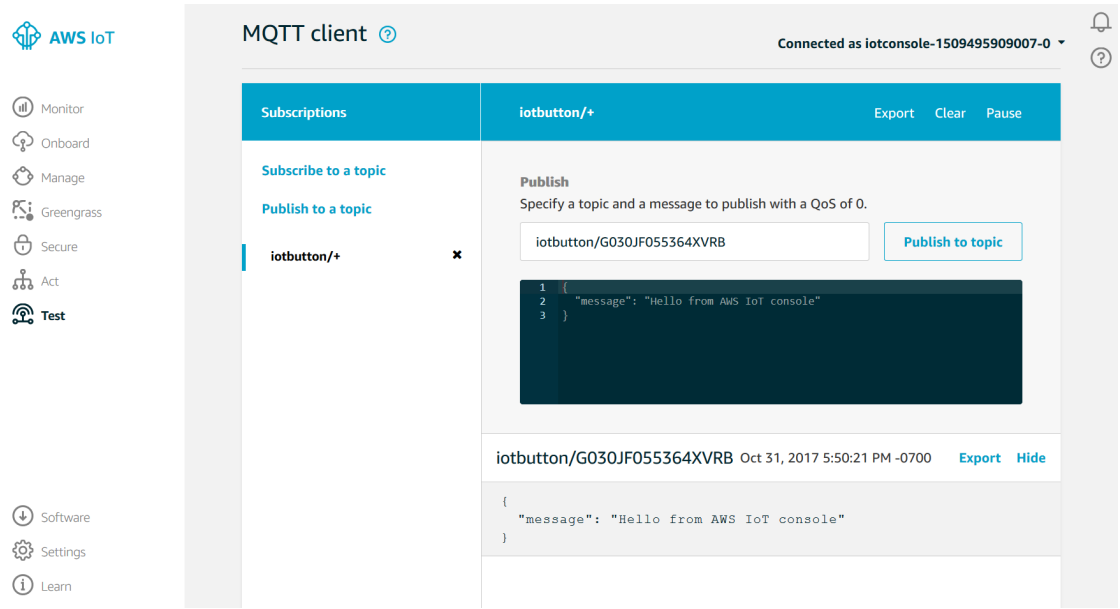
AWS IoT MQTT Client

To test your rule with the AWS IoT MQTT client:

1. In the AWS IoT console, in the left navigation pane, choose **Test**.
2. On the MQTT client page, in the **Publish** section, in the **Specify a topic and a message to publish**... field, type **`my/topic`** or the topic you used in the rule. In the message payload section, type the following JSON:

```
{
    "message": "Hello, from AWS IoT console"
}
```

Note
If you are using a button, type **`iotbutton/`*`your-button-DSN`*** instead of **`my/topic`** in the **Specify a topic and a message to publish**... field.



3. Choose **Publish to topic**. You should receive an Amazon SNS message on your cell phone.

Congratulations! You have successfully created and configured a rule that sends data received from a device to an Amazon SNS topic.

Next Steps

For more information about AWS IoT rules, see AWS IoT Rule Tutorials (p. 46) and AWS IoT Rules (p. 162).

AWS IoT Button Quickstarts

The two quickstarts in this section show you how to configure and use the AWS IoT button. You can use the AWS IoT button wizard in the AWS Lambda console to easily and quickly configure your AWS IoT button. The AWS Lambda console contains a blueprint that automates the process of setting up your AWS IoT button by:

- Creating and activating an X.509 certificate and private key for authenticating with AWS IoT.
- Walking you through the configuration of your AWS IoT button in order to connect to your Wi-Fi network.
- Walking you through the copying of your certificate and private key to your AWS IoT button.
- Creating and attaching to the certificate an AWS IoT policy that gives the button permission to make calls to AWS IoT.
- Creating an AWS IoT rule that invokes a Lambda function when your AWS IoT button is pressed.
- Creating an IAM role and policy that allows the Lambda function to send email messages using Amazon SNS.
- Creating a Lambda function that sends an email message to the address specified in the Lambda function code.

The second quickstart shows you how to use an AWS CloudFormation template to configure the AWS IoT resources required to process the MQTT messages that are sent when the AWS IoT button is pressed.

If you do not have a button, you can purchase one here. For more information about AWS IoT, see What Is AWS IoT (p. 1).

Topics

AWS IoT Button Wizard Quickstart

The AWS IoT button wizard is a Lambda blueprint, so you must sign in to the AWS Lambda console in order to use it. If you do not have an AWS account, you can create one by following these steps.

To create an AWS account

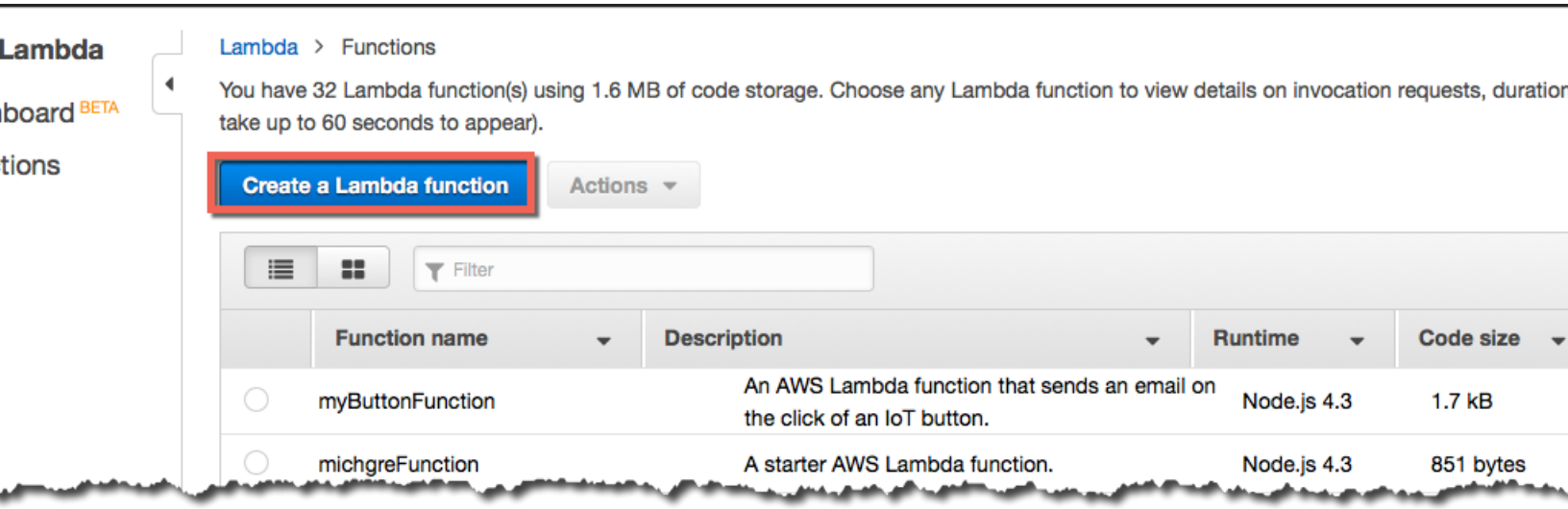
1. Open the AWS home page and choose **Create an AWS Account**.

2. Follow the online instructions. Part of the sign-up procedure involves receiving a phone call and entering a PIN using your phone's keypad.

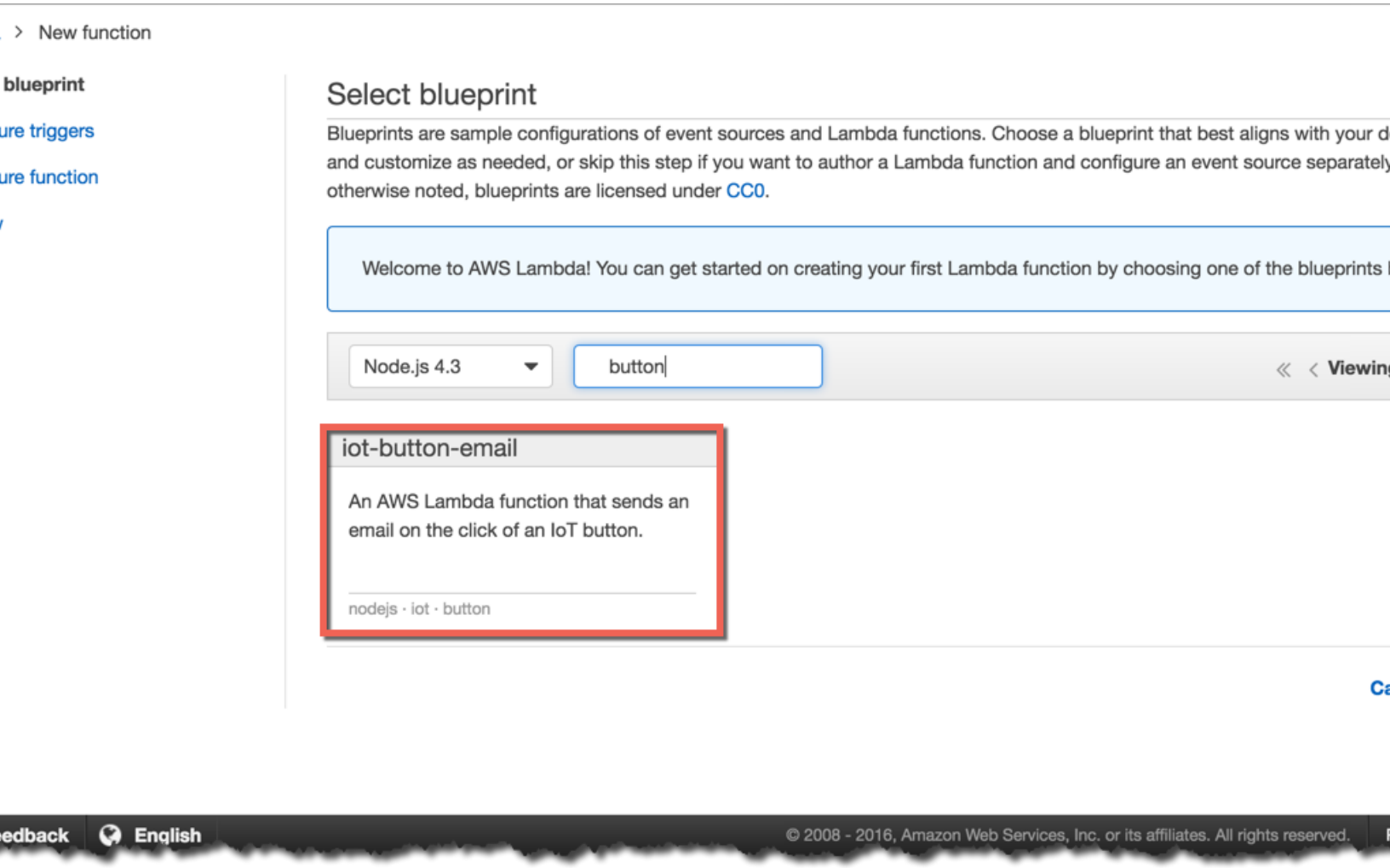
To configure the AWS IoT Button

1. Sign in to the AWS Management Console and open the AWS Lambda console.
2. If this is your first time in the AWS Lambda console, you see the following page. Choose the **Get Started Now** button.

If you have used the AWS Lambda console before, you see the following page. Choose the **Create a Lambda function** button.



3. On the **Select blueprint** page, from the **Runtime** drop-down menu, choose **Node.js 4.3**. In the filter text box, type **`button`**. To choose the **iot-button-email** blueprint, double-click it or choose the **Next** button.

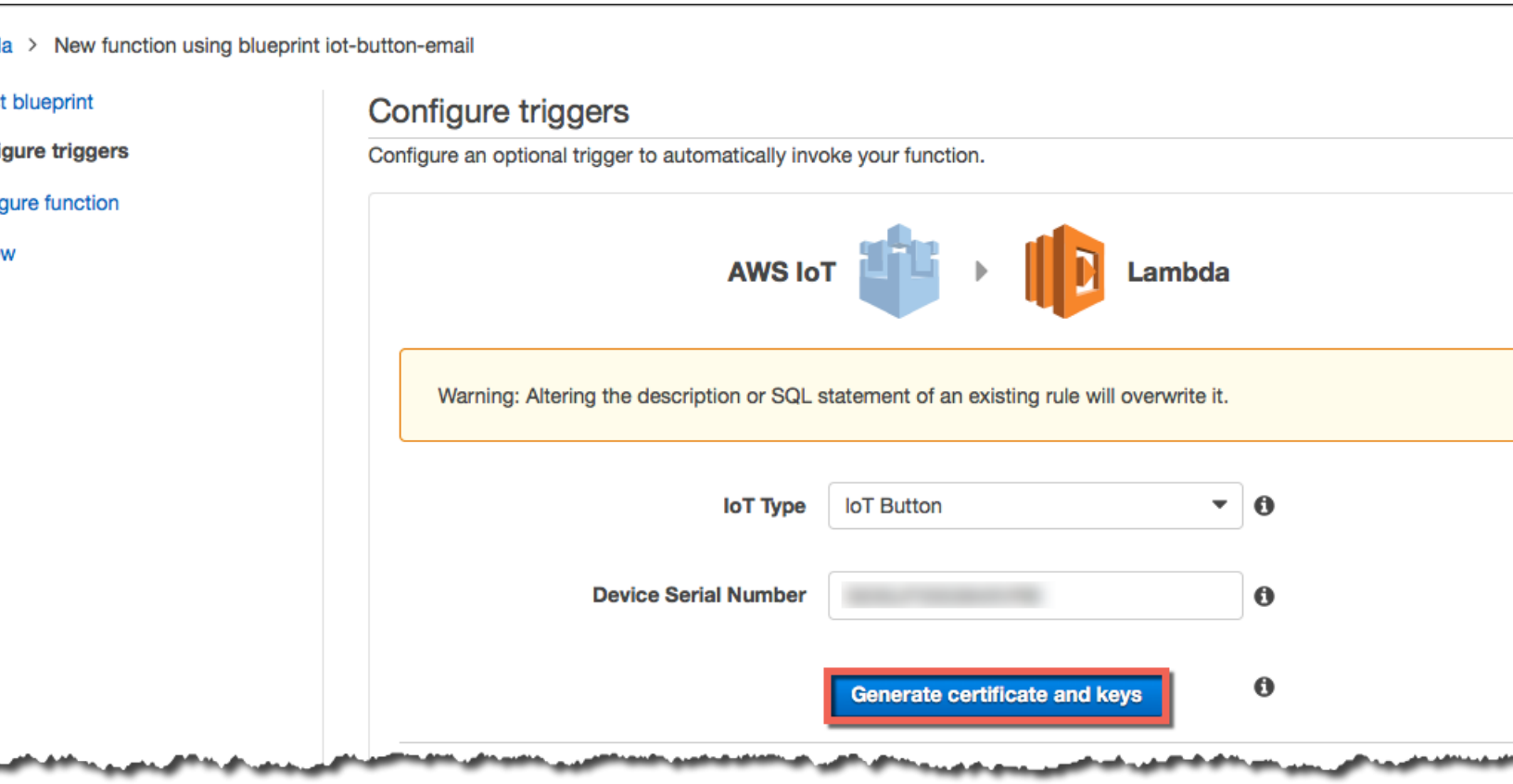


4. On the **Configure triggers** page, from the **IoT Type** drop-down menu, choose **IoT Button**.

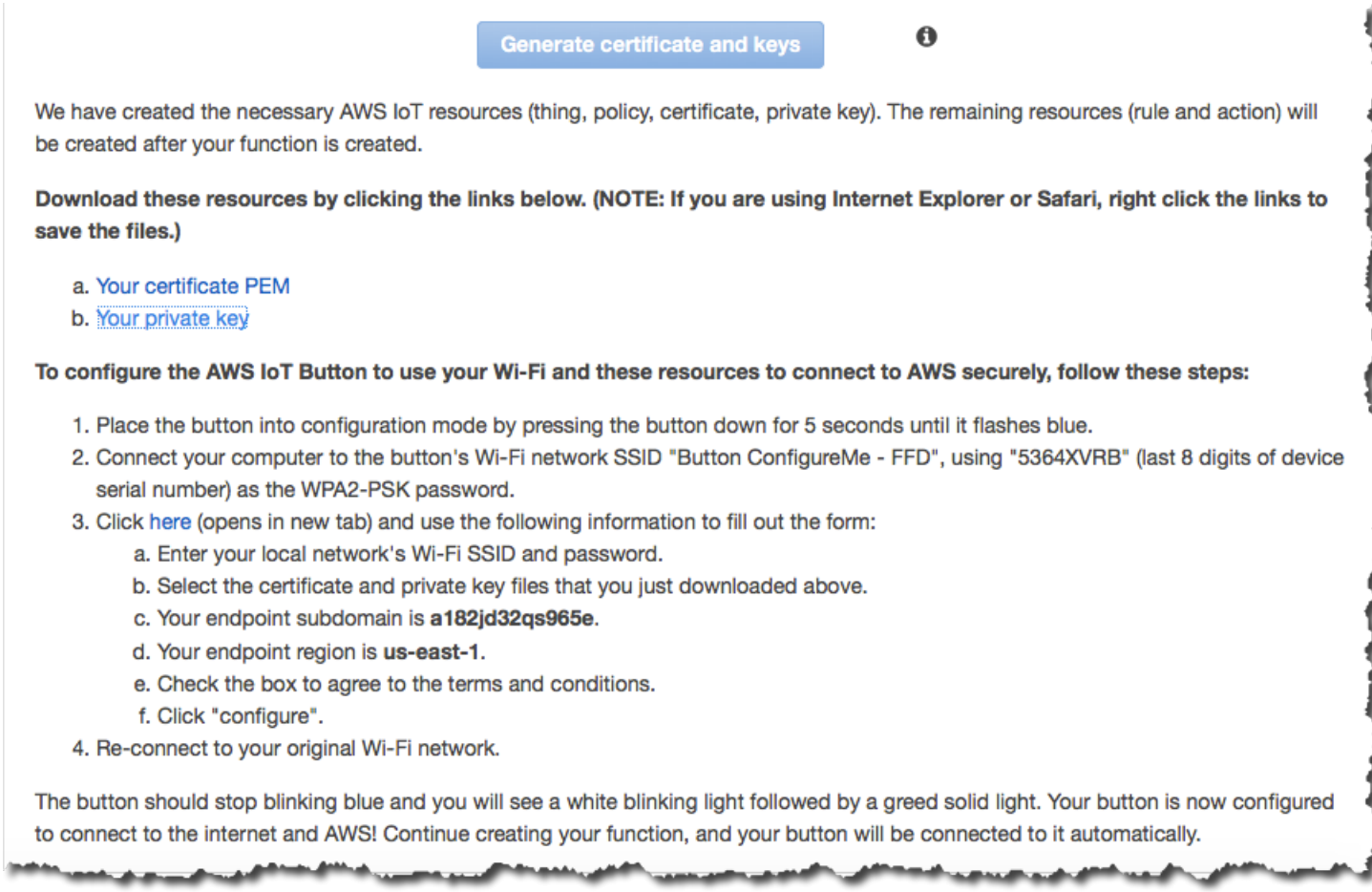
   Type the serial number for your device. The device serial number (DSN) appears on the back of the button.

   Choose **Generate certificate and keys**.

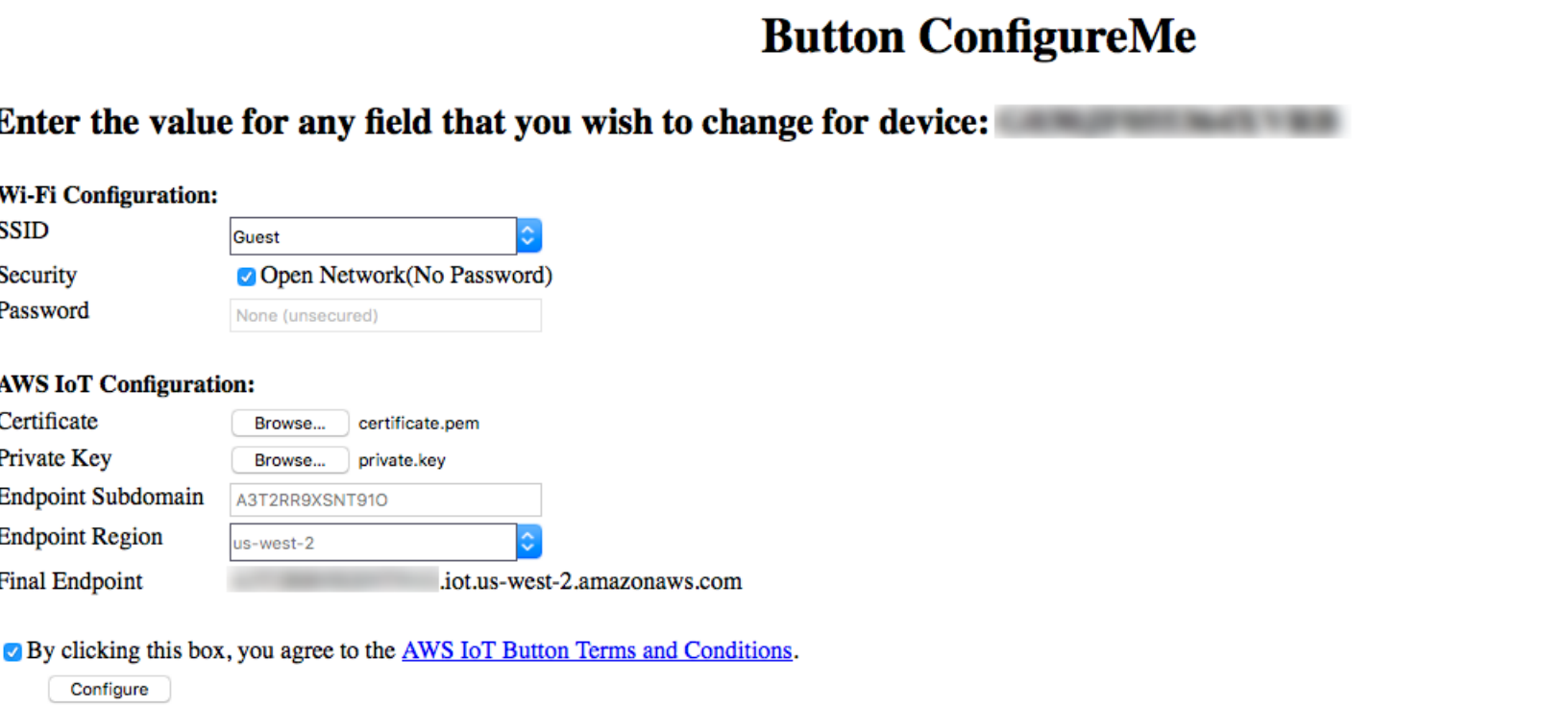
   Note
   You only need to generate a certificate and private key once. Then you can navigate to http://192.168.0.1/index.html in a browser to configure your button.



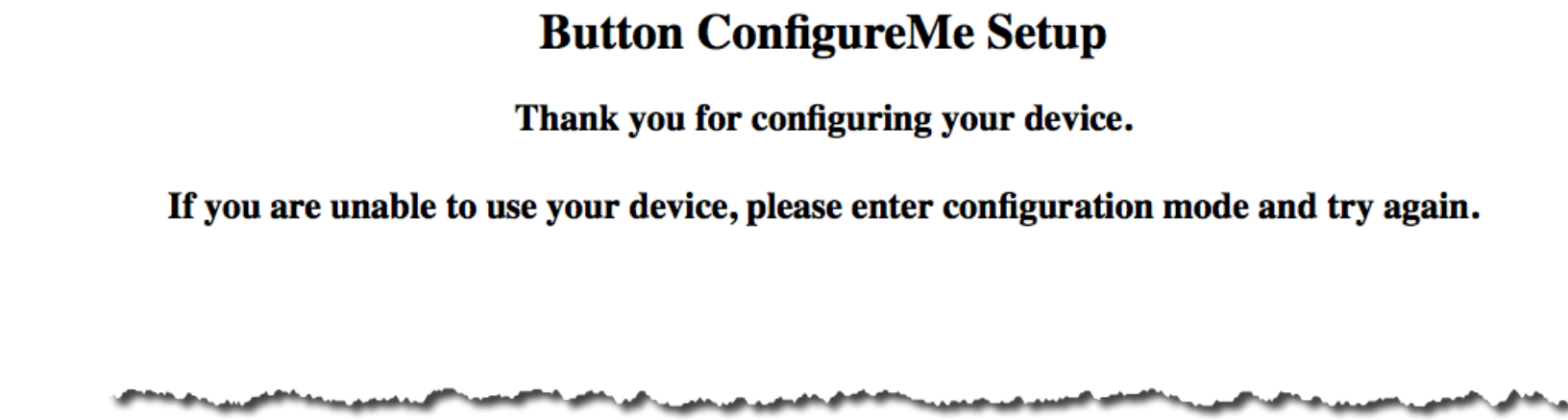
Use the links on the page to download the device certificate and the private key.



The page also includes instructions for configuring your AWS IoT button. On step 3, you choose a link to open a web page that allows you to connect the AWS IoT button to your network. Under **Wi-Fi Configuration**, type the network ID (SSID) and network password for your Wi-Fi network. Under **AWS IoT Configuration**, choose the certificate and private key you downloaded earlier. This copies your certificate and private key to your AWS IoT button. Select the check box to agree to the AWS IoT button terms and conditions, and then choose the **Configure** button.

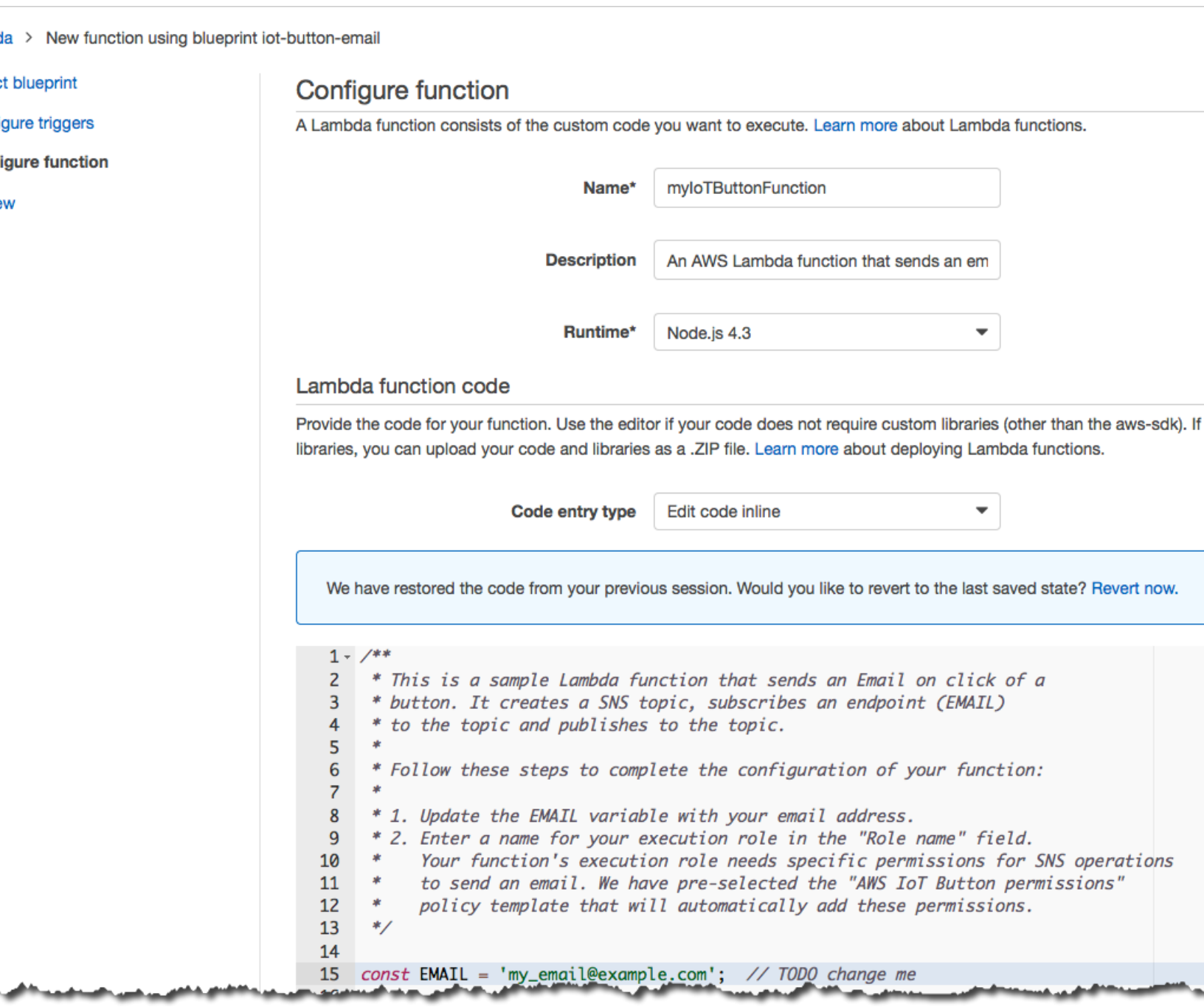


A configuration confirmation page is displayed.



5. Close the **Configure** tab and go back to the AWS Lambda console page. Choose **Enable trigger**, and then choose **Next**.

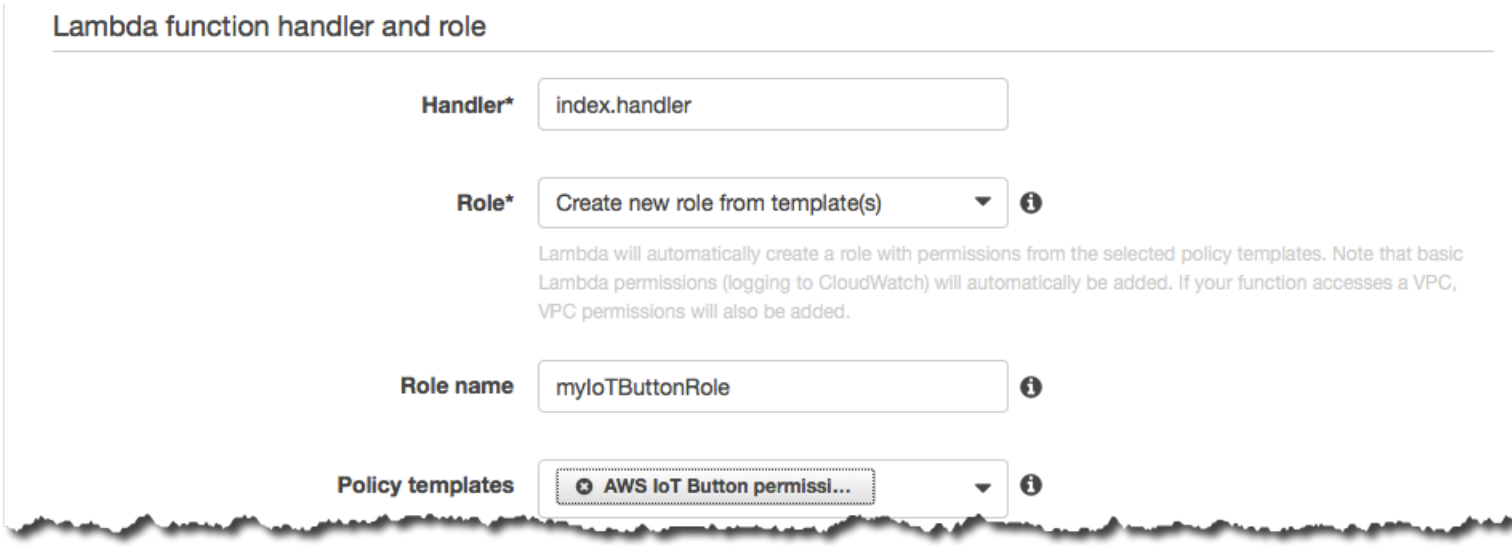
   On the **Configure function** page, type a name for your function. The description, runtime, and Lambda function code is entered for you.



In the Lambda function code, replace the example email address with your own email address.

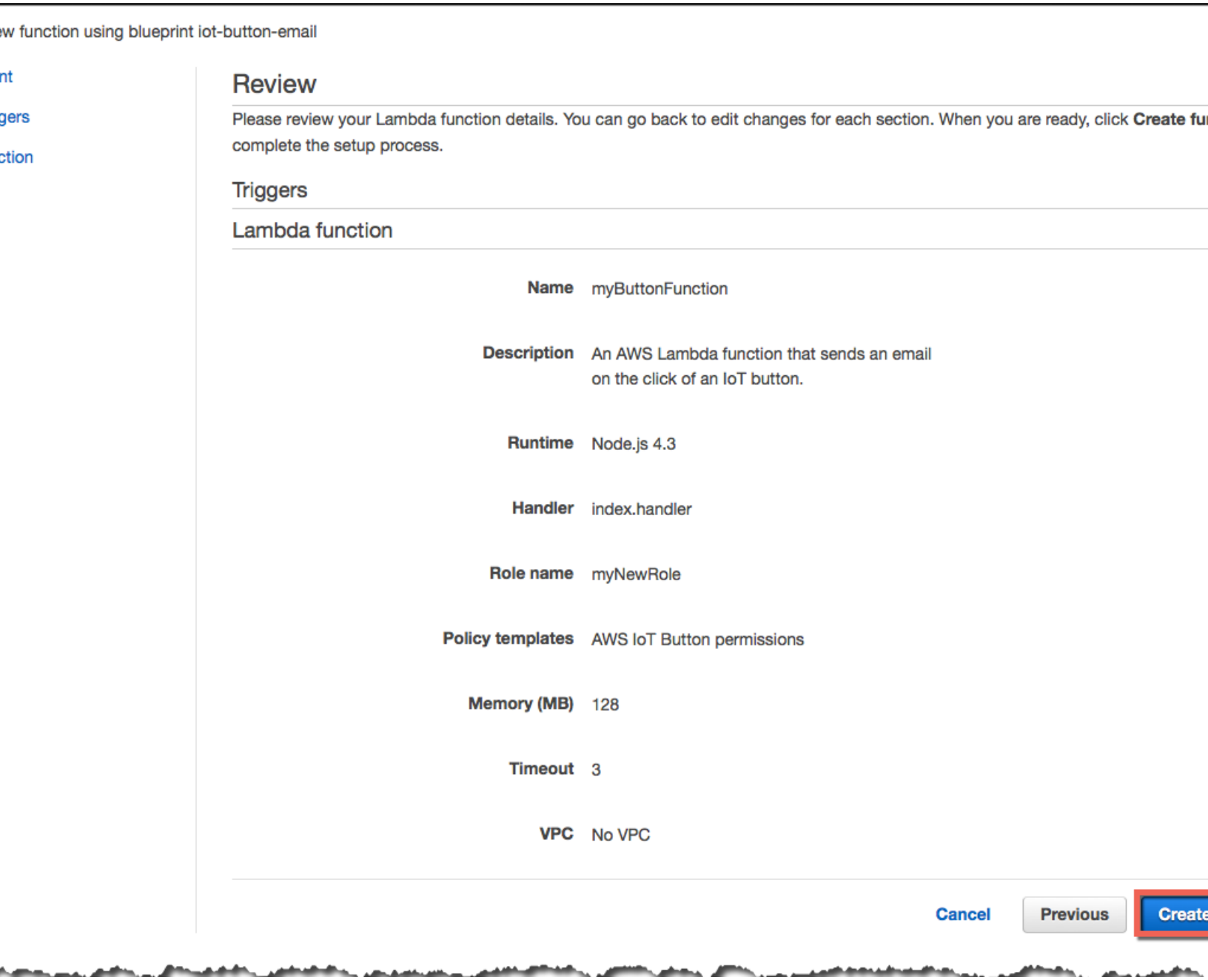
```
/**
 * This is a sample Lambda function that sends an Email on click of a
 * button. It creates a SNS topic, subscribes an endpoint (EMAIL)
 * to the topic and publishes to the topic.
 *
 * Follow these steps to complete the configuration of your function:
 *
 * 1. Update the EMAIL variable with your email address.
 * 2. Enter a name for your execution role in the "Role name" field.
 *    Your function's execution role needs specific permissions for SNS operations
 *    to send an email. We have pre-selected the "AWS IoT Button permissions"
 *    policy template that will automatically add these permissions.
 */

const EMAIL = 'my_email@example.com';  // TODO change me

const AWS = require('aws-sdk');
const SNS = new AWS.SNS({ apiVersion: '2010-03-31' });

function findExistingSubscription(topicArn, nextToken, cb) {
    const params = {
        TopicArn: topicArn,
        NextToken: nextToken || null,
    };
    SNS.listSubscriptionsByTopic(params, (err, data) => {
        if (err) {
```

In the **Lambda function handler and role** section, from the **Role** drop-down menu, choose **Create new role from template(s)**. Type a unique name for the role.

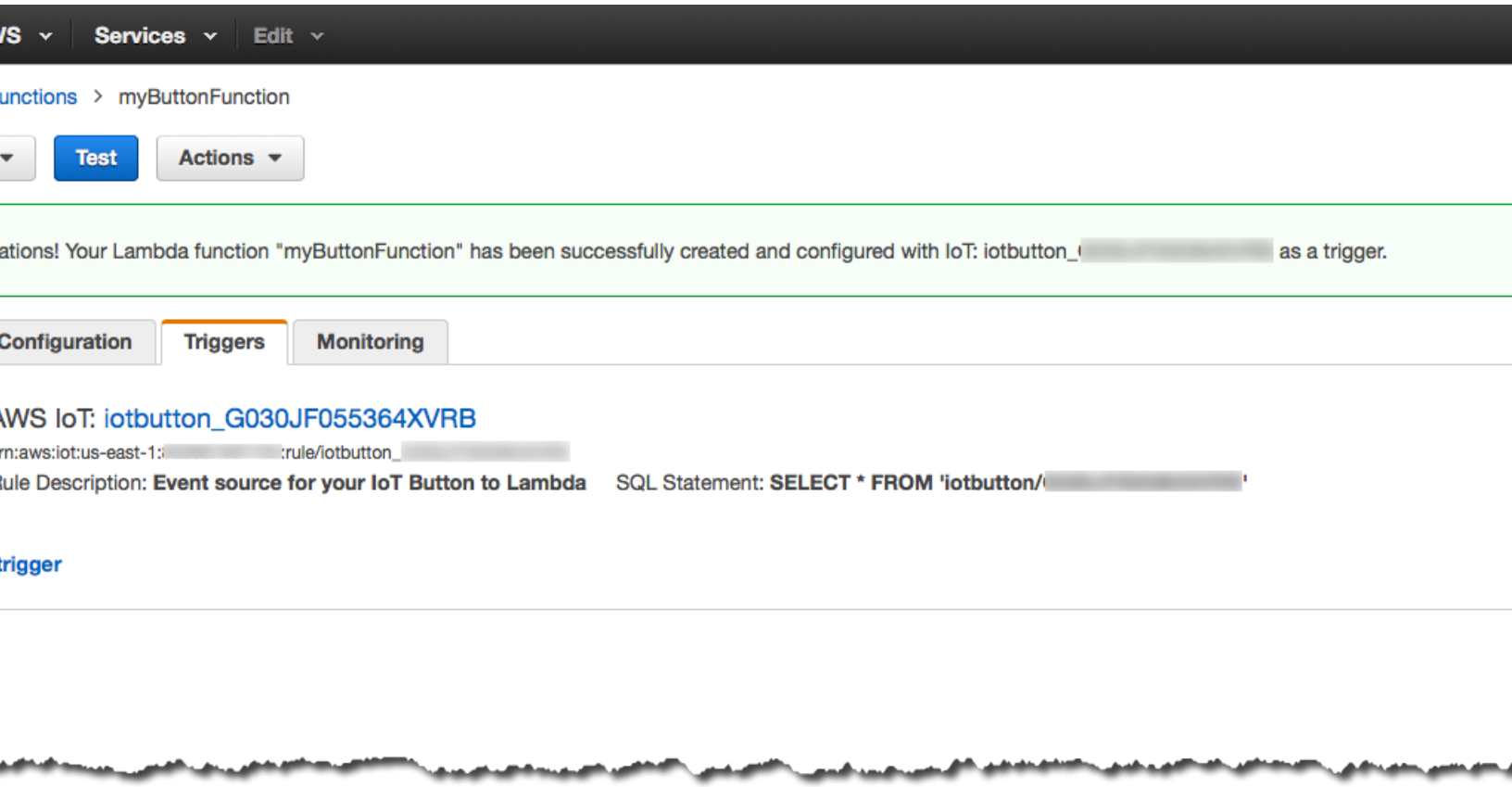


At the bottom of the page, choose **Next**.

Review the settings for the Lambda function, and then choose **Create function**.



You should see a page that confirms your Lambda function has been created:



6. To test your Lambda function, choose the **Test** button. After about a minute, you should receive an email message with `AWS Notification - Subscription Confirmation` in the subject line. Choose the link in the email message to confirm the subscription to an SNS topic created by the Lambda function. When AWS IoT receives a message from your button, it sends a message to Amazon SNS. The Lambda function created a subscription to the Amazon SNS topic using the email address you added in the code. When Amazon SNS receives a message on this Amazon SNS topic, it forwards the message to your subscribed email address.

Press your button to send a message to AWS IoT. The message causes your Lambda rule to be triggered, and then your Lambda function is invoked. The Lambda function checks if your SNS topic exists. The Lambda function then sends the contents of the message to the Amazon SNS topic. Amazon SNS then forwards the message to the email address you specified in the Lambda function code.

AWS IoT Button AWS CloudFormation Quickstart

When the AWS IoT button is pressed, it sends basic information about the button to an Amazon SNS topic. The topic then forwards that information to you in an email message. This quickstart shows you how to use an AWS CloudFormation template to configure your AWS IoT button.

You need an AWS account and an AWS IoT button to complete the steps in this quickstart.

1. Use the AWS IoT console to create an AWS IoT certificate:

   a. Open the AWS IoT console.
   b. If a **Welcome** page appears, choose **Get started**.

c. In the AWS region selector, choose the AWS region where you want to create the AWS IoT certificate (for example, US East (N. Virginia)). You will be creating all supporting AWS resources (additional AWS IoT resources and an Amazon SNS resource) in the same AWS region.

d. On the **Dashboard**, in the left navigation pane, choose **Security**, and then choose **Certificates**.

e. On the **Certificates** pane, choose **Create**.

f. Choose **One-click certificate creation** - **Create certificate**.

g. On the **Certificate created** page, choose **Download** for the certificate, private key, and root CA for AWS IoT, save each of them to your computer, and then choose **Activate**.

h. Choose **Done**.

i. On the **Certificates** page, choose the certificate you just created.

j. In the **Details** pane, make a note of the certificate ARN value (for example, `arn:aws:iot:region-ID:account-ID:cert/random-ID`). You need it later in this procedure.

2. Use the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/ to create the AWS IoT resources, an Amazon SNS resource, and an IAM role:

   a. Save the following AWS CloudFormation template file named AWSIoTButtonQuickStart.template to your computer.

      Note
      Check the DSN for your button. Make sure the beginning of the "AllowedPattern" string matches your DSN.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Creates required AWS resources to allow an AWS IoT button to send
 information through an Amazon Simple Notification Service (Amazon SNS) topic to an
 email address.",
  "Parameters": {
    "IoTButtonDSN": {
   "Type": "String",
   "AllowedPattern": "G030[A-Z][A-Z][0=9][0-9][0-9][0-5][0-9][1-7][0-9A-HJ-NP-X]
[0-9A-HJ-NP-X][0-9A-HJ-NP-X][0-9A-HJ-NP-X]",
   "Description": "The device serial number (DSN) of the AWS IoT Button. This can
 be found on the back of the button. The DSN must match the pattern of 'G030[A-Z]
[A-Z][0=9][0-9][0-9][0-5][0-9][1-7][0-9A-HJ-NP-X][0-9A-HJ-NP-X][0-9A-HJ-NP-X][0-9A-
HJ-NP-X]'."
 },
 "CertificateARN": {
   "Type": "String",
   "Description": "The Amazon Resource Name (ARN) of the existing AWS IoT
 certificate."
 },
 "SNSTopicName": {
   "Type": "String",
   "Default": "aws-iot-button-sns-topic",
   "Description": "The name of the Amazon SNS topic for AWS CloudFormation to
 create."
 },
 "SNSTopicRoleName": {
   "Type": "String",
   "Default": "aws-iot-button-sns-topic-role",
   "Description": "The name of the IAM role for AWS CloudFormation to create. This
 IAM role allows AWS IoT to send notifications to the Amazon SNS topic."
 },
 "EmailAddress": {
   "Type": "String",
   "Description": "The email address for the Amazon SNS topic to send information
 to."
 }
```

```
 },
 "Resources": {
   "IoTThing": {
     "Type": "AWS::IoT::Thing",
     "Properties": {
       "ThingName": {
   "Fn::Join" : [ "",
     [
       "iotbutton_",
             { "Ref": "IoTButtonDSN" }
  ]
   ]
 }
     }
   },
"IoTPolicy": {
     "Type" : "AWS::IoT::Policy",
     "Properties": {
       "PolicyDocument": {
   "Version": "2012-10-17",
         "Statement": [
           {
             "Action": "iot:Publish",
             "Effect": "Allow",
             "Resource": {
      "Fn::Join": [ "",
           [
             "arn:aws:iot:",
    { "Ref": "AWS::Region" },
    ":",
    { "Ref": "AWS::AccountId" },
    ":topic/iotbutton/",
    { "Ref": "IoTButtonDSN" }
        ]
         ]
    }
           }
         ]
       }
     }
   },
"IoTPolicyPrincipalAttachment": {
     "Type": "AWS::IoT::PolicyPrincipalAttachment",
     "Properties": {
       "PolicyName": {
   "Ref": "IoTPolicy"
 },
 "Principal": {
   "Ref": "CertificateARN"
 }
     }
   },
"IoTThingPrincipalAttachment": {
     "Type" : "AWS::IoT::ThingPrincipalAttachment",
     "Properties": {
       "Principal": {
   "Ref": "CertificateARN"
 },
 "ThingName": {
   "Ref": "IoTThing"
 }
     }
   },
"SNSTopic": {
     "Type": "AWS::SNS::Topic",
  "Properties": {
```

```
    "DisplayName": "AWS IoT Button Press Notification",
    "Subscription": [
   {
     "Endpoint": {
    "Ref": "EmailAddress"
  },
  "Protocol": "email"
   }
 ],
       "TopicName": {
   "Ref": "SNSTopicName"
 }
    }
   },
"SNSTopicRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
   "Version": "2012-10-17",
         "Statement": [
           {
             "Effect": "Allow",
             "Principal": {
               "Service": "iot.amazonaws.com"
             },
             "Action": "sts:AssumeRole"
           }
         ]
 },
    "Path": "/",
 "Policies": [
   {
     "PolicyDocument": {
    "Version": "2012-10-17",
             "Statement": [
      {
                 "Effect": "Allow",
                 "Action": "sns:Publish",
                 "Resource": {
       "Fn::Join": [ "",
               [
                 "arn:aws:sns:",
        { "Ref": "AWS::Region" },
        ":",
        { "Ref": "AWS::AccountId" },
        ":",
        { "Ref": "SNSTopicName" }
                     ]
          ]
           }
   }
             ]
  },
  "PolicyName": {
    "Ref": "SNSTopicRoleName"
  }
   }
 ]
  }
},
"IoTTopicRule": {
     "Type": "AWS::IoT::TopicRule",
     "Properties": {
       "RuleName": {
   "Fn::Join": [ "",
     [
```

```
        "iotbutton_",
        { "Ref": "IoTButtonDSN" }
    ]
     ]
  },
  "TopicRulePayload": {
    "Actions": [
      {
        "Sns": {
        "RoleArn": {
          "Fn::GetAtt": [ "SNSTopicRole", "Arn" ]
        },
        "TargetArn": {
          "Ref": "SNSTopic"
        }
      }
    }
     ],
     "AwsIotSqlVersion": "2015-10-08",
           "RuleDisabled": false,
           "Sql": {
       "Fn::Join": [ "",
         [
           "SELECT * FROM 'iotbutton/",
     { "Ref": "IoTButtonDSN" },
     "'"
         ]
    ]
     }
         }
       }
     }
  }
}
```

b. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.

c. Make sure the region where you created the AWS IoT certificate (for example, US East (N. Virginia)) is displayed on the AWS region selector.

d. Choose **Create Stack**.

e. On the **Select Template** page, choose **Upload a template to Amazon S3**, and then choose **Browse**.

f. Select the AWSIoTButtonQuickStart.template file you saved earlier, choose **Open**, and then choose **Next**.

g. On the **Specify Details** page, for **Stack name**, type a name for this AWS CloudFormation stack (for example, MyAWSIoTButtonStack).

h. For **CertificateARN**, type the Amazon Resource Name (ARN) of the AWS IoT certificate (the certificate ARN value) that you noted earlier.

i. For **EmailAddress**, type your email address.

j. For **IoTButtonDSN**, type the device serial number (DSN). It appears on the back of your AWS IoT button (for example, G030JF051234A5BC).

k. You can leave **SNSTopicName** and **SNSTopicRoleName** at their defaults, or specify a different Amazon SNS topic name and associated IAM role name. For example, if you plan to set up more AWS IoT buttons, you might want to change these values. Choose **Next**.

l. You do not need to do anything on the **Options** page. Choose **Next**.

m. On the **Review** page, select **I acknowledge that AWS CloudFormation might create IAM resources**, and then choose **Create**.

   n. When CREATE_COMPLETE is displayed for MyAWSIoTButtonStack, check your email inbox for a message with a subject line of AWS IoT Button Press Notification. Choose the **Confirm subscription** link in the body of the email message.

3. Using the private key and certificate you created earlier, follow the steps in Configure Your Device to set up your AWS IoT button.
4. After you have set it up, press the button once. A white light should blink several times and then be followed by a steady green light for a few moments. Shortly afterward, you should receive an email message with AWS IoT Button Press Notification in the subject line. It contains information sent by the button in the body of the email message.
5. After you are finished experimenting, you can clean up the AWS resources created by the AWS CloudFormation template. To do this, return to the AWS CloudFormation console and delete MyAWSIoTButtonStack. After you delete MyAWSIoTButtonStack, delete the AWS IoT certificate as follows:

   a. Return to the AWS IoT console.
   b. In the list of resources, select the check box inside of the box that represents the AWS IoT certificate (the box with the handshake icon).
   c. For **Actions**, choose **Deactivate**, and then confirm.
   d. With the box that represents the AWS IoT certificate still selected, for **Actions**, choose **Delete**, and then confirm.
   e. The private key and certificate that you downloaded earlier are no longer valid, so you can now delete them from your computer.

Next Steps

To learn more about the Lambda blueprint used to set up your button, see Getting Started with AWS IoT. To learn how to use AWS CloudFormation with the AWS IoT button, see http://docs.aws.amazon.com/iot/latest/developerguide/iot-button-cloud-formation.html.

AWS IoT Rule Tutorials

This guide includes tutorials that walk you through the creation and testing of AWS IoT rules. If you have not completed the AWS IoT Getting Started Tutorial (p. 5), we recommend you do that first. It shows you how to create an AWS account and connect your device to AWS IoT.

An AWS IoT rule consists of a SQL SELECT statement, a topic filter, and a rule action. Devices send information to AWS IoT by publishing messages to MQTT topics. The SQL SELECT statement allows you to extract data from an incoming MQTT message. The topic filter of an AWS IoT rule specifies one or more MQTT topics. The rule is triggered when an MQTT message is received on a topic that matches the topic filter. Rule actions allow you to take the information extracted from an MQTT message and send it to another AWS service. Rule actions are defined for AWS services like Amazon DynamoDB, AWS Lambda, Amazon SNS, and Amazon S3. By using a Lambda rule, you can call other AWS or third-party web services. For a complete list of rule actions, see AWS IoT Rule Actions (p. 172).

In these tutorials we assume you are using the AWS IoT button and will use `iotbutton/+` as the topic filter in the rules. If you do not have an AWS IoT button, you can buy one here.

Alternatively, you can emulate the AWS IoT button by using an MQTT client like the AWS IoT MQTT client in the AWS IoT console. To emulate the AWS IoT button, publish a similar message on the `iotbutton/ABCDEFG12345` topic. The number after the / is arbitrary. It is used as the serial number for the button.

You can also use your own device, but you must know on which MQTT topic your device publishes so you can specify it as the topic filter in the rule. For more information, see AWS IoT Rules (p. 162).

The AWS IoT button sends a JSON payload that looks like this:

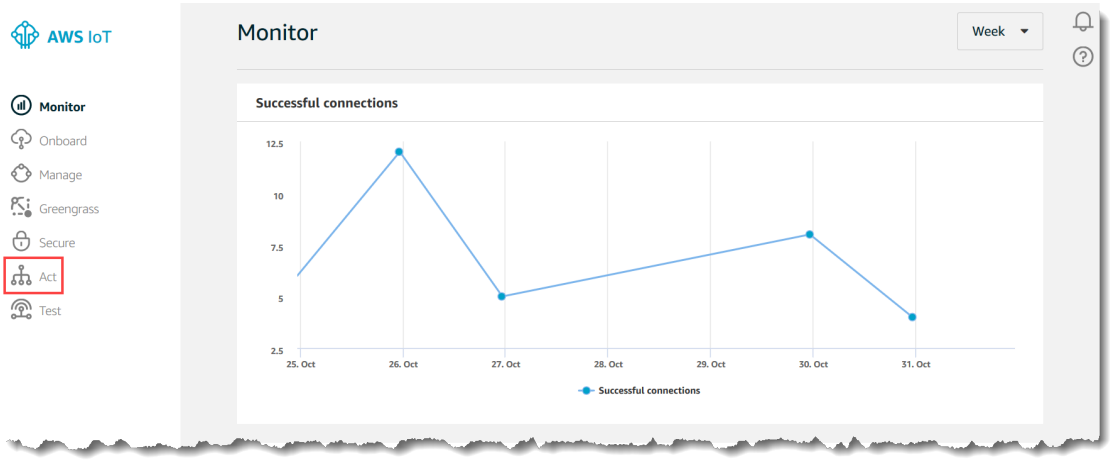
```
{
    "serialNumber" : "ABCDEFG12345",
    "batteryVoltage" : "2000mV",
    "clickType" : "SINGLE"
}
```

Creating a DynamoDB Rule

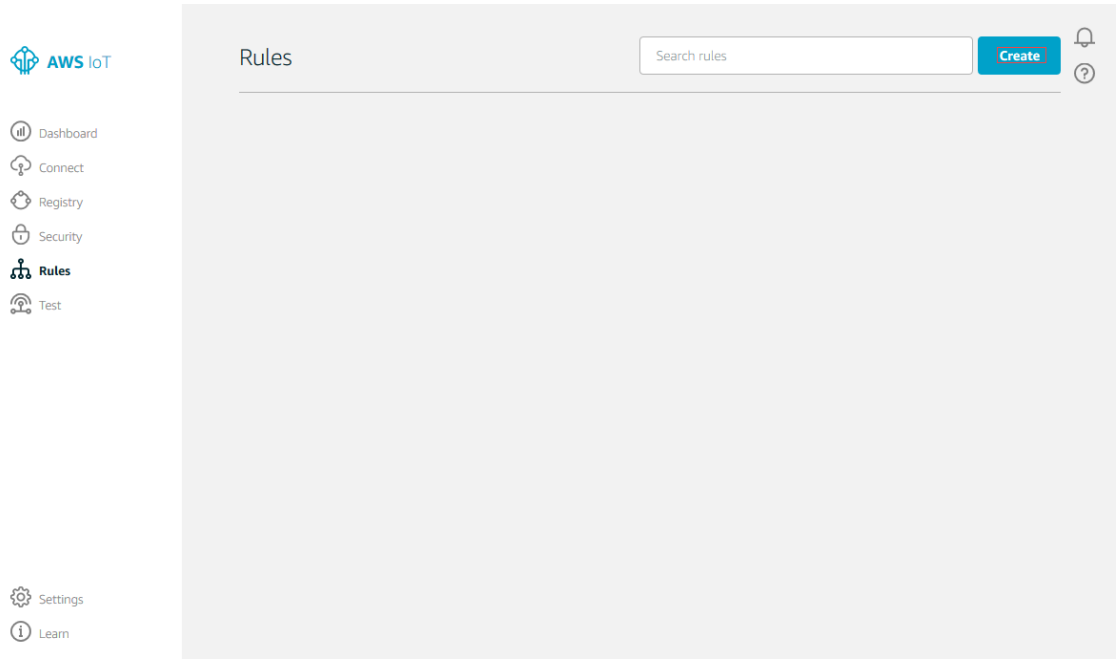
DynamoDB rules allow you to take information from an incoming MQTT message and write it to a DynamoDB table.

To create a DynamoDB rule:

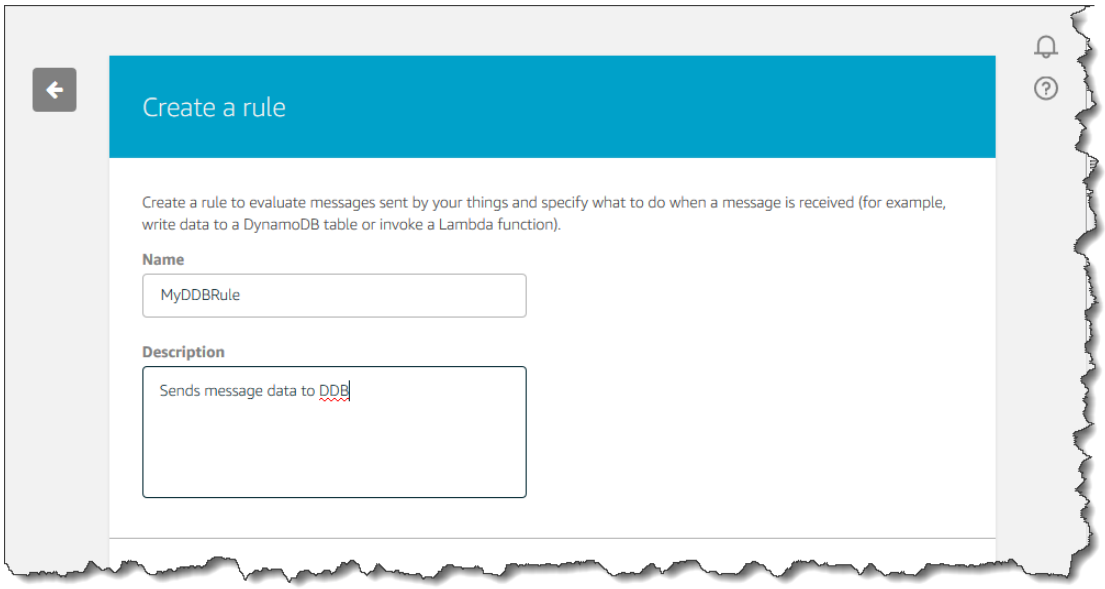
1. In the AWS IoT console, in the left navigation pane, choose **Rules**.



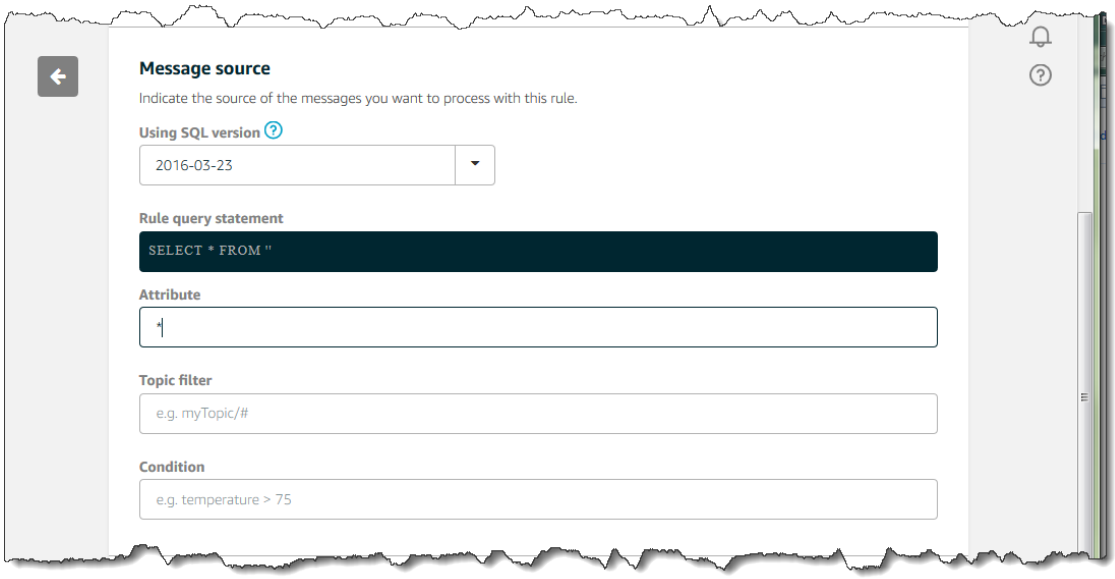
2. On the **Rules** page, choose **Create**.



3. On the **Create a rule** page, in the **Name** field, type a name for your rule. In the **Description** field, type a description for the rule.



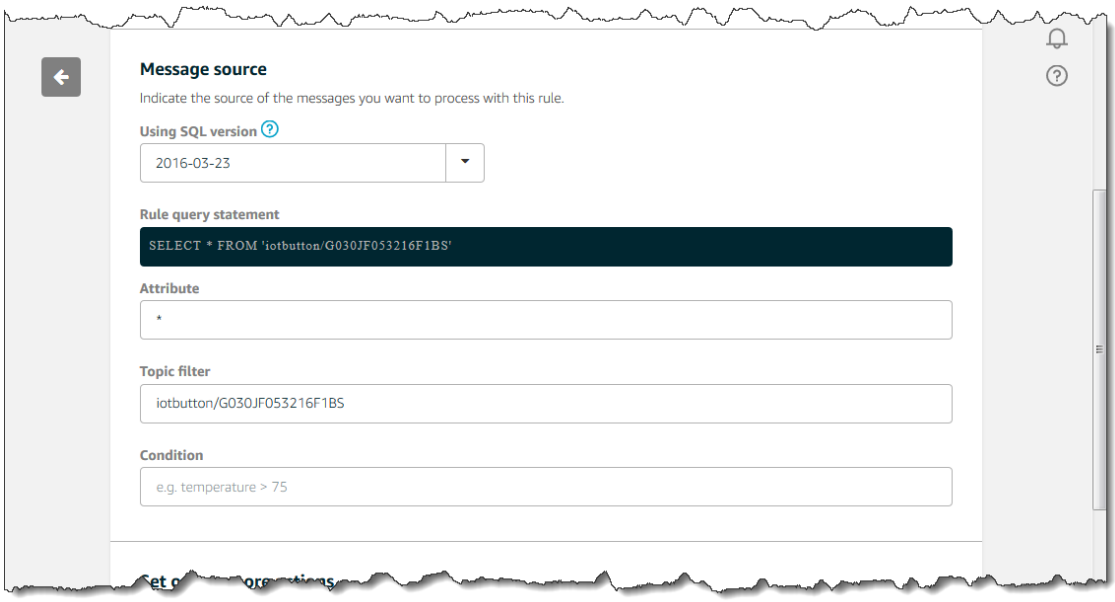
4. Scroll down to **Message source**. Choose the latest version from the **Using SQL version** drop-down list. In the **Attribute** field, type *. This specifies that you want to send the entire MQTT message that triggered the rule.



5. The rules engine uses the topic filter to determine which rules to trigger when an MQTT message is received. In the **Topic filter** field, type **`iotbutton/`*`your-button-DSN`***. If you are not using an AWS IoT button, type **`my/topic`** or the topic used in the rule.

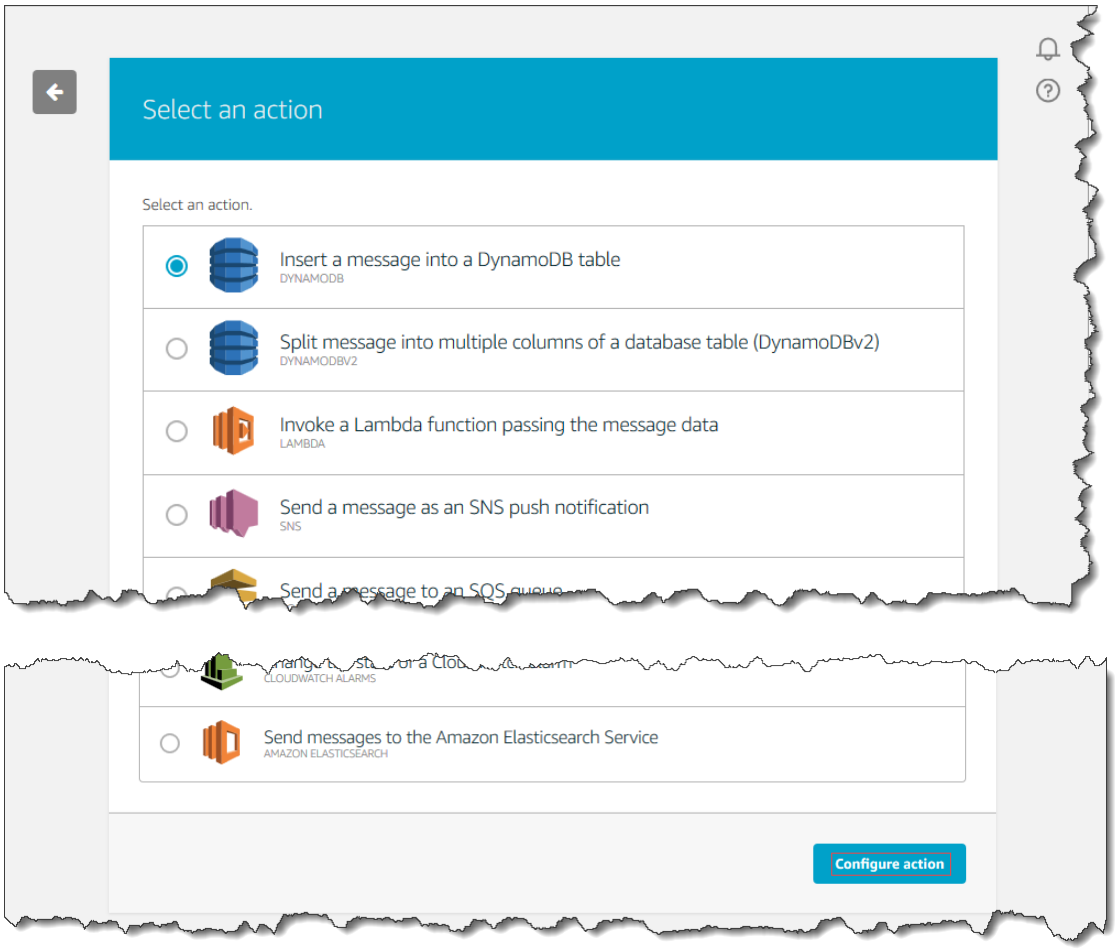
   Note
   You can find the DSN on the bottom of the button.

   Leave **Condition** blank.

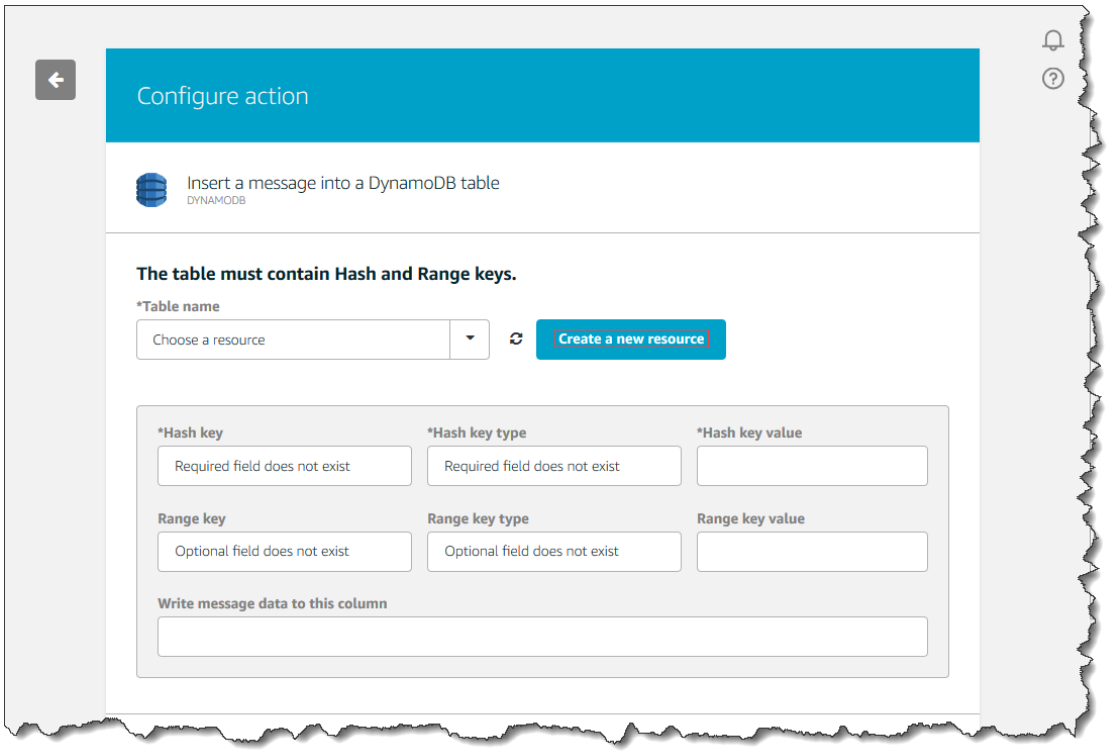


6. In **Set one or more actions**, choose **Add action**.

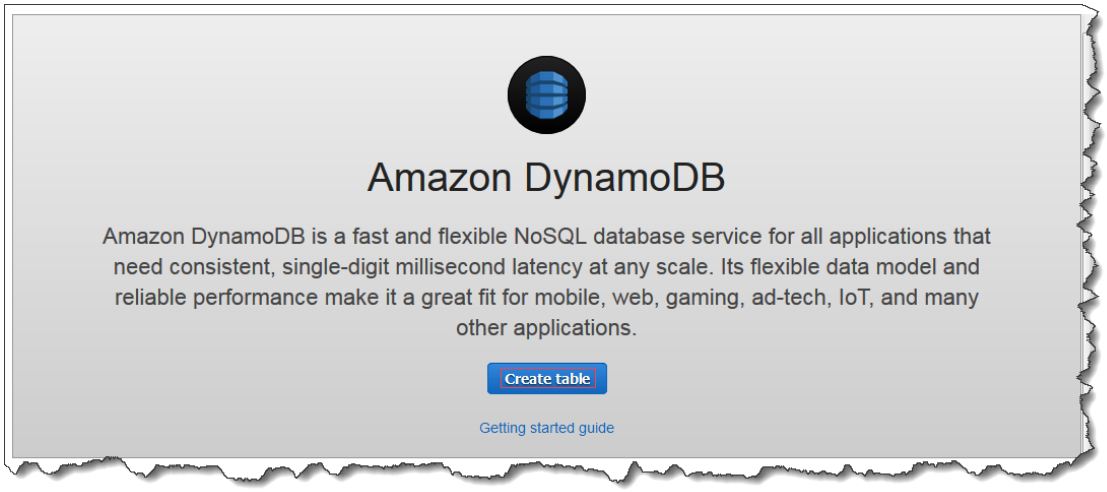
7. On the **Select an action** page, select **Insert a message into a DynamoDB table**, and then choose **Configure action**.



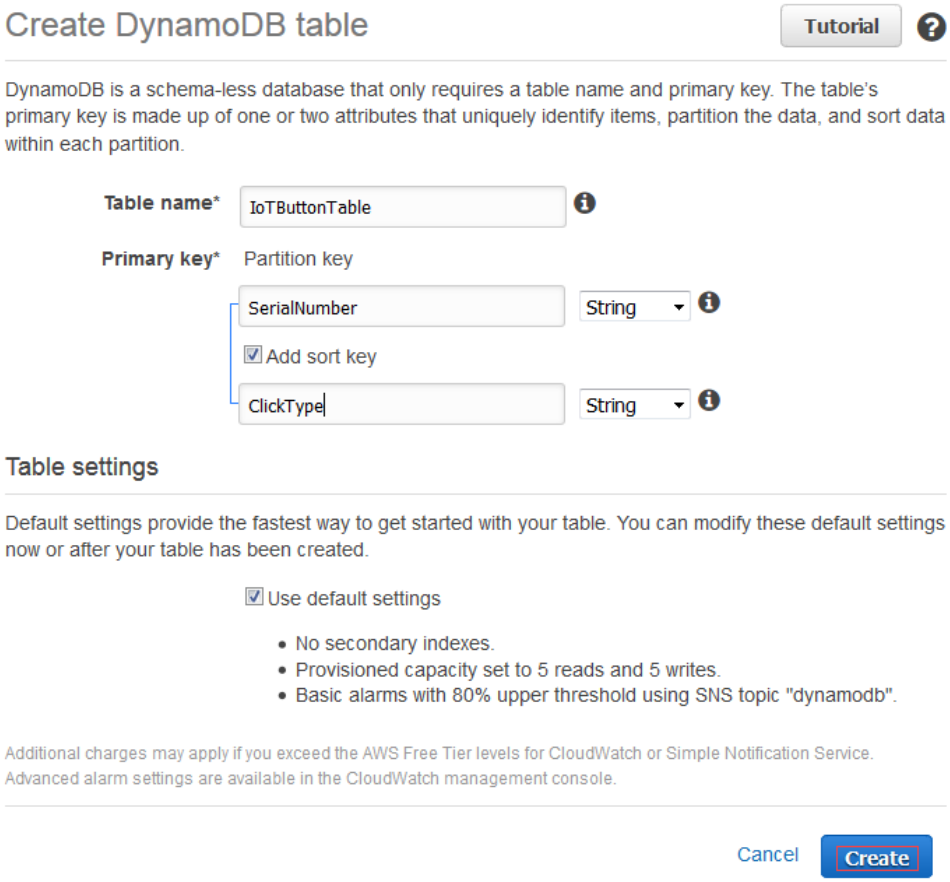
8. On the **Configure action** page, choose **Create a new resource**.



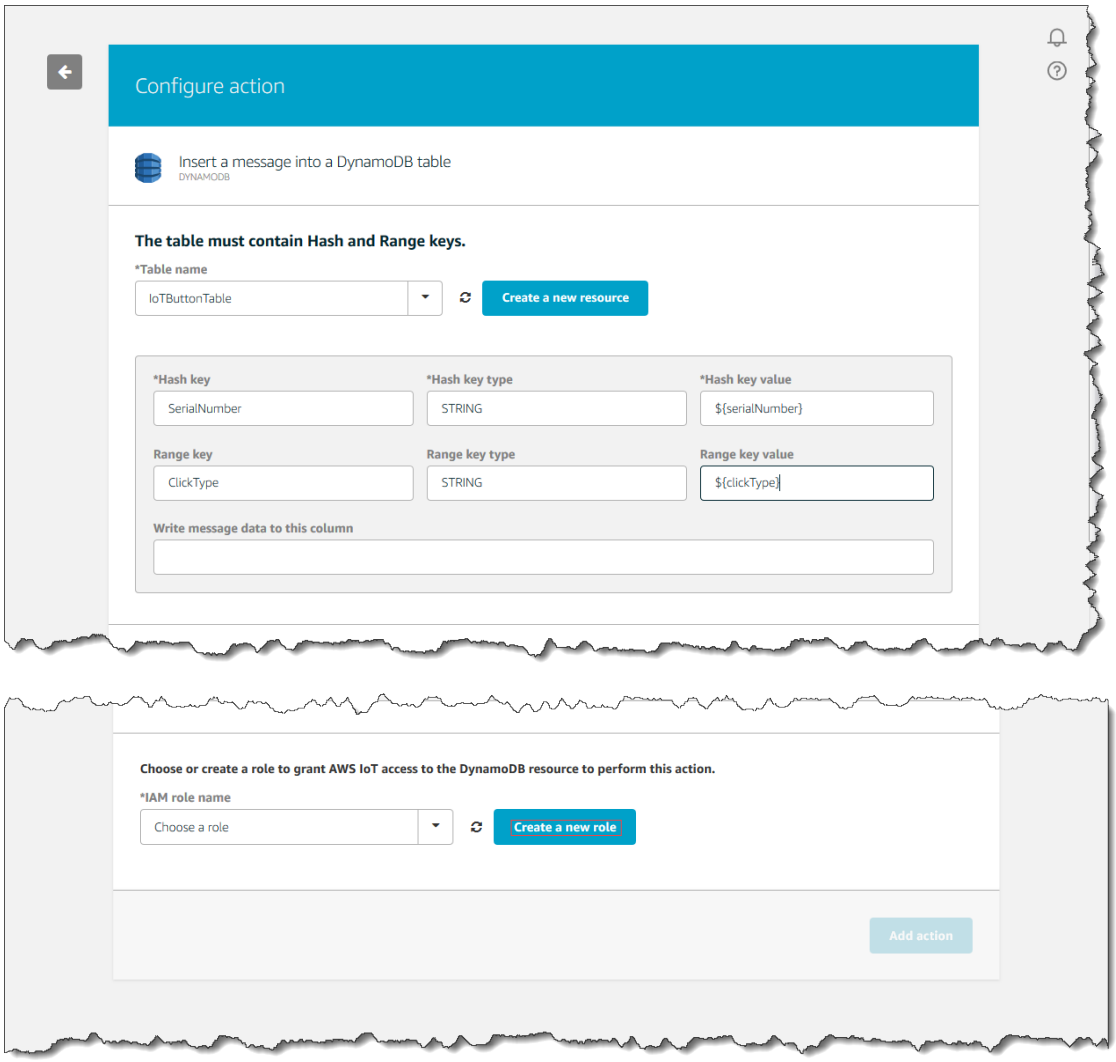
9. On the **Amazon DynamoDB** page, choose **Create table**.



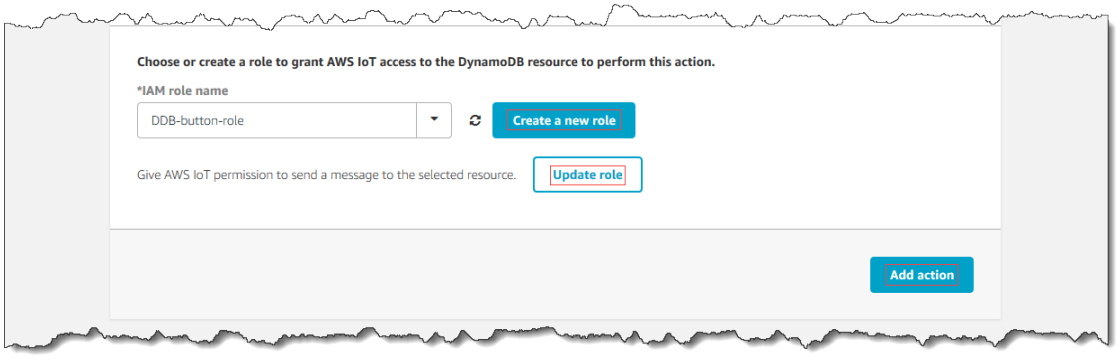
10. On the **Create DynamoDB table** page, type a name in the **Table name** field. In **Partition key**, type **`SerialNumber`**. Select the **Add sort key** check box, and then type **`ClickType`** in the **Sort key** field. Select **String** for both the partition and sort keys.



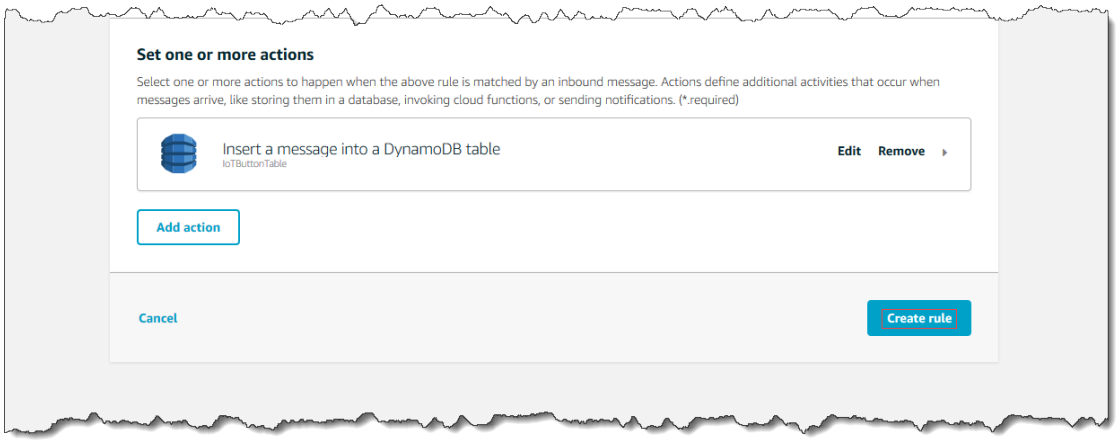
11. Choose **Create**. It takes a few seconds to create your DynamoDB table. Close the browser tab where the Amazon DynamoDB console is open. If you do not close the tab, your DynamoDB table will not be displayed in the **Table name** drop-down list on the AWS IoT **Configure action** page.
12. On the **Configure action** page, choose your new table from the **Table name** drop-down list. In **Hash key value**, type **`${serialNumber}`**. This instructs the rule to take the value of the `serialNumber` attribute from the MQTT message and write it into the **SerialNumber** column in the DynamoDB table. In **Range key value**, type **`${clickType}`**. This writes the value of the `clickType` attribute into the **ClickType** column. Leave **Write message data to this column** blank. By default, the entire message is written to a column in the table named Payload. Choose **Create a new role**.



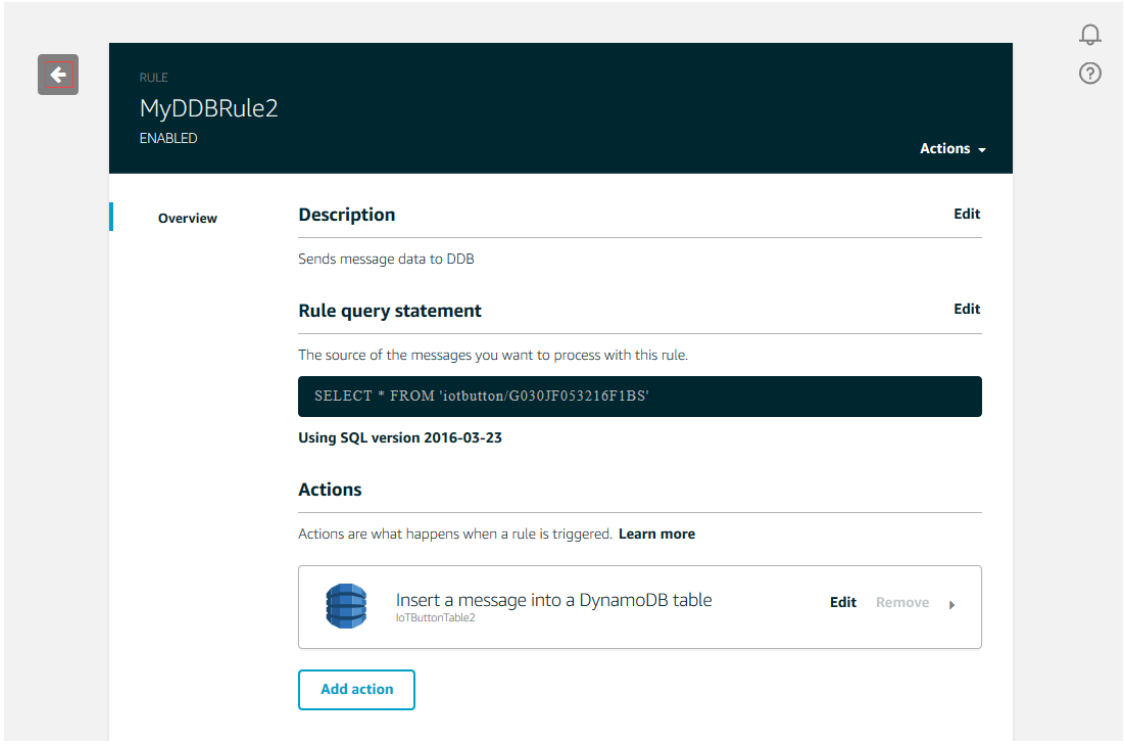
13. Type a unique name in **IAM role name**, and then choose the **Create a new role** button again. Choose the role you just created, choose **Update role**, and then choose **Add action**.



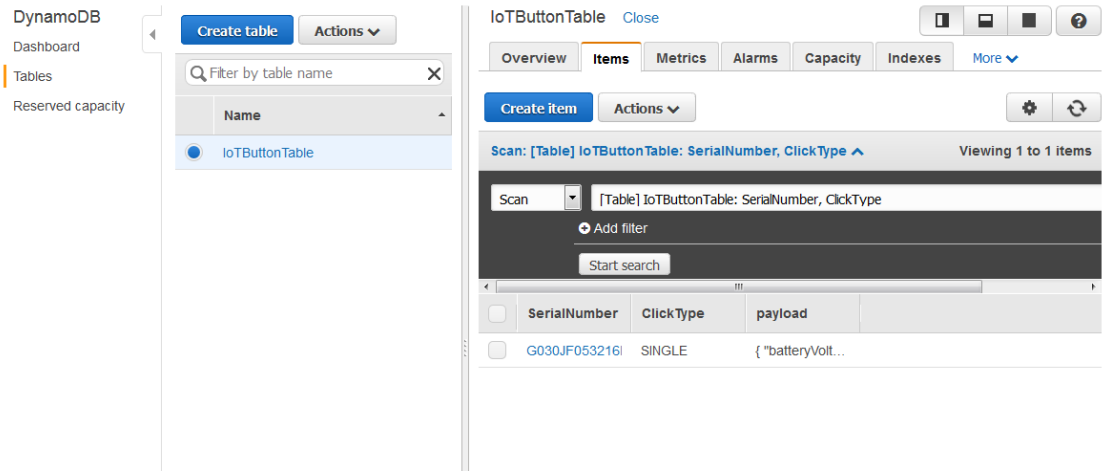
14. Choose **Create rule** to create the rule.



15. A confirmation message shows the rule has been created. Choose the left arrow to return to the **Rules** page.



16. Test the rule by pressing your configured AWS IoT button or by using an MQTT client to publish a message on a topic that matches your rule's topic filter. Finally, return to the DynamoDB console and select the table you created to view the entry for your button press or message.



Creating a Lambda Rule

You can define a rule that calls a Lambda function, passing in data from the MQTT message that triggered the rule. This allows you to process the incoming message and then call another AWS or third-party service.

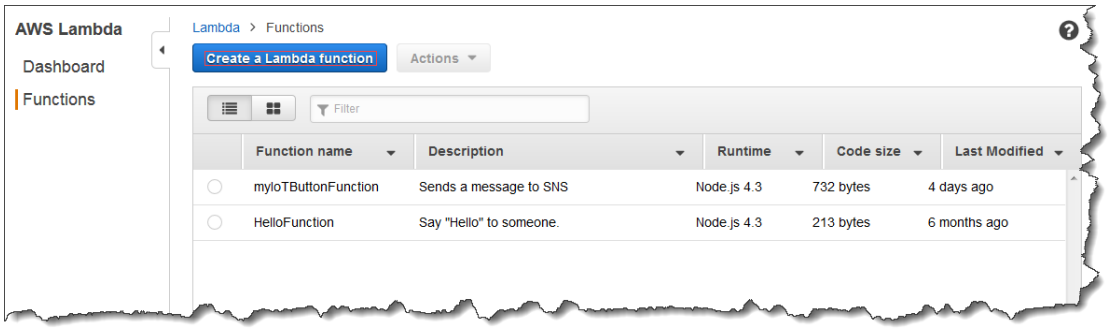
In this tutorial, we assume you have completed the AWS IoT Getting Started Tutorial (p. 5) in which you create and subscribe to an Amazon SNS topic using your cell phone number. You will create a Lambda function that publishes a message to the Amazon SNS topic you created in the AWS IoT Getting Started Tutorial (p. 5). You will also create a Lambda rule that calls the Lambda function, passing in some data from the MQTT message that triggered the rule.

In this tutorial, we also assume you are using an AWS IoT button to trigger the Lambda rule. If you do not have an AWS IoT button, you can buy one here or you can use an MQTT client to send an MQTT message that triggers the rule.

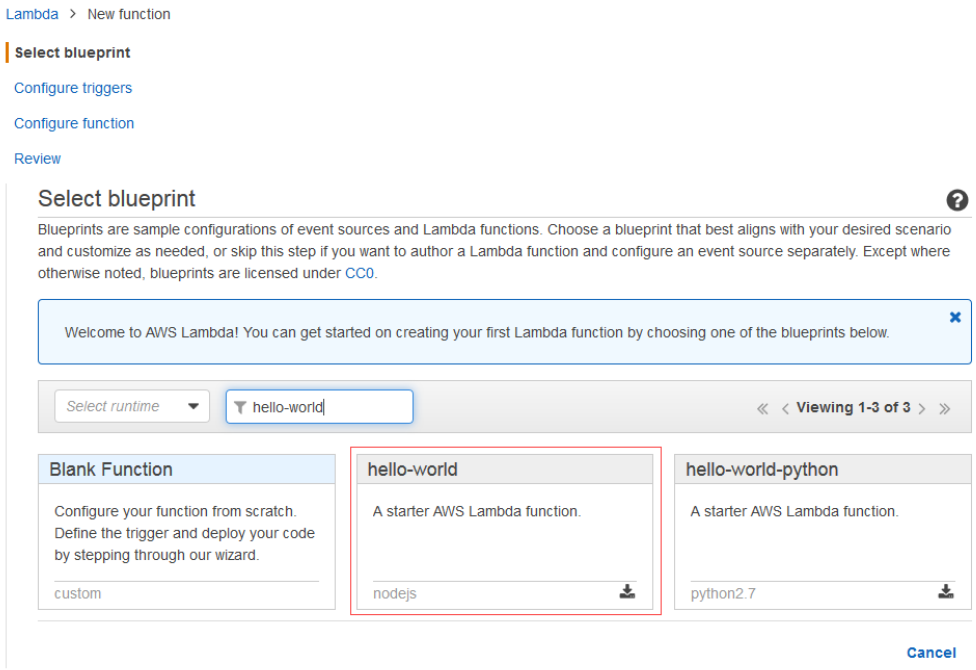
Create the Lambda Function

To create the Lambda function:

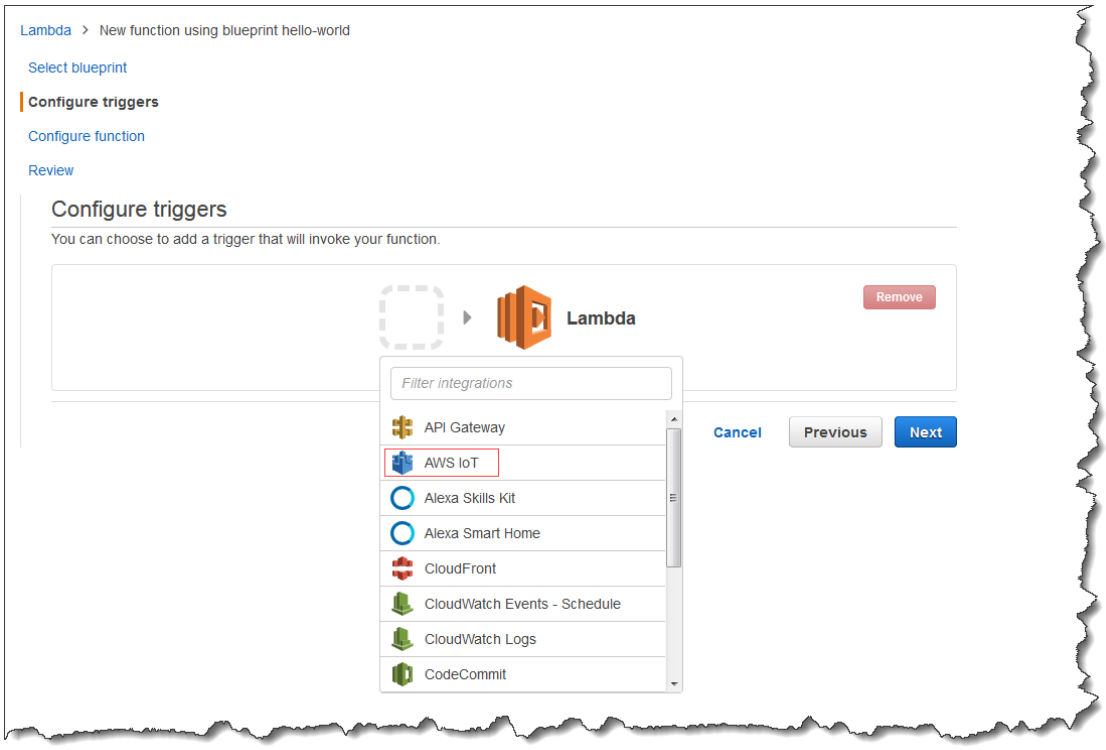
1. In the AWS Lambda console, choose **Get Started Now** or, if you have created a Lambda function before, choose **Create a Lambda function**.



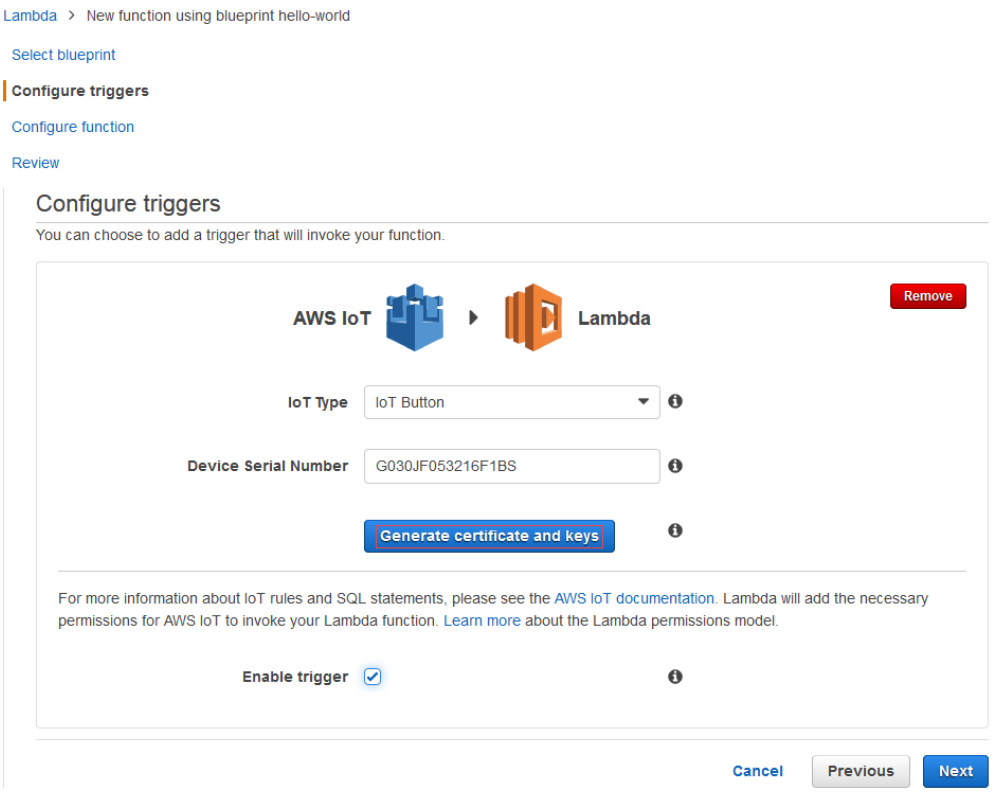
2. On the **Select blueprint** page, in the **Filter** field, type **`hello-world`**, and then choose the **hello-world** blueprint.



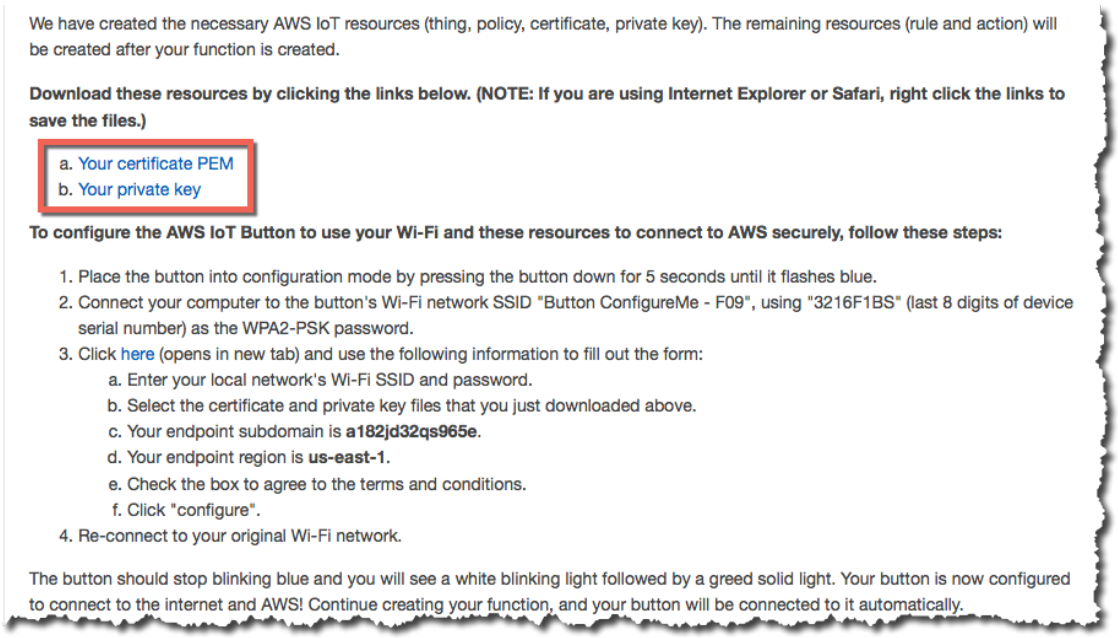
3. On the **Configure triggers** page, select the box to the left of the Lambda icon, and select **AWS IoT** from the drop-down menu.



4. In the **Device Serial Number** field, type your button's device serial number (DSN). The DSN is printed on the back of your AWS IoT button. If you have not already generated a certificate and private key for your AWS IoT button, choose **Generate certificate and keys**. Otherwise, skip to step 6.

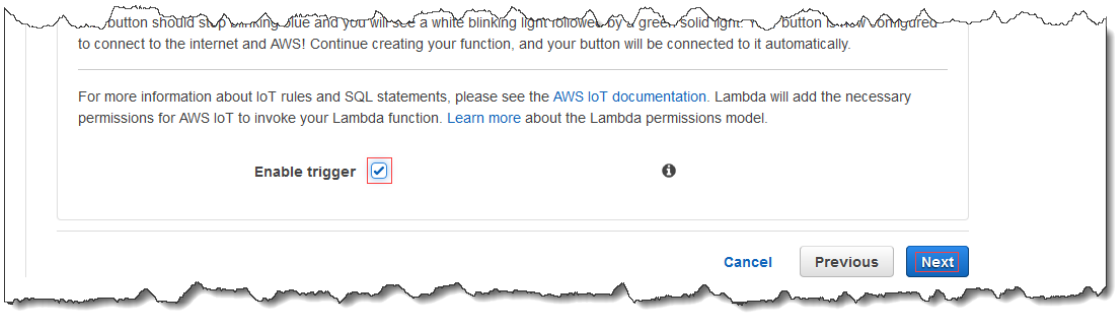


5. Choose the links to download your certificate PEM and private key. Save these files in a secure location on your computer.

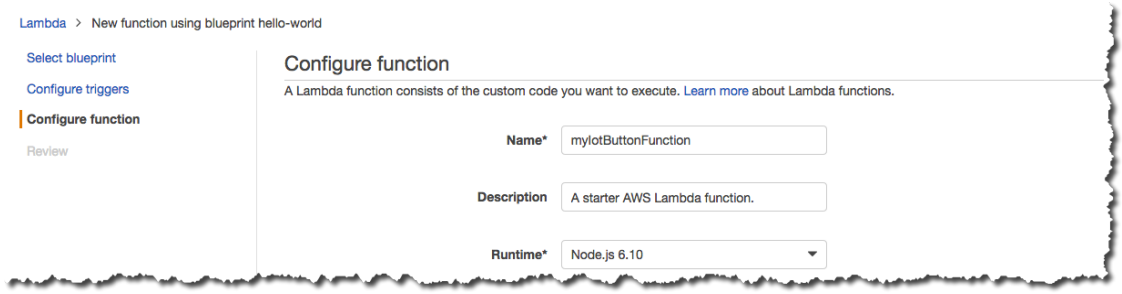


Follow the online instructions to configure your AWS IoT button.

6. Make sure that the **Enable trigger** check box is selected, and then choose **Next**.



7. On the **Configure function** page, type a name and description for the Lambda function. In **Runtime**, choose **Node.js 6.10**.



8. Scroll down to the **Lambda function code** section of the page. Replace the existing code with the following:

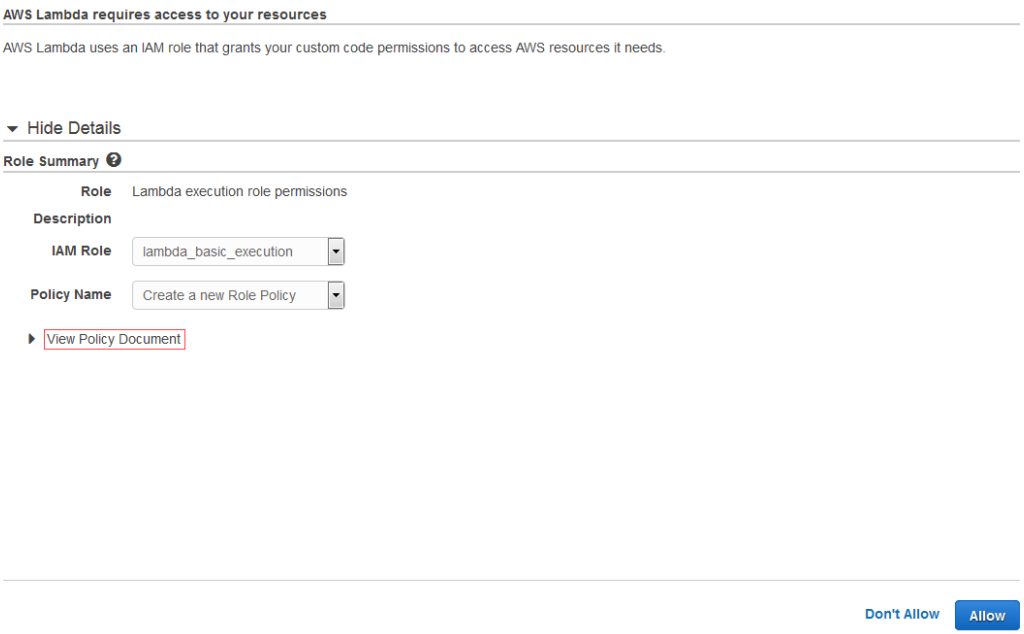
```
console.log('Loading function');
    // Load the AWS SDK
    var AWS = require("aws-sdk");

    // Set up the code to call when the Lambda function is invoked
    exports.handler = (event, context, callback) => {
        // Load the message passed into the Lambda function into a JSON object
        var eventText = JSON.stringify(event, null, 2);

        // Log a message to the console, you can view this text in the Monitoring tab
 in the Lambda console or in the CloudWatch Logs console
        console.log("Received event:", eventText);

        // Create a string extracting the click type and serial number from the message
 sent by the AWS IoT button
        var messageText = "Received  " + event.clickType + " message from button ID: "
 + event.serialNumber;

        // Write the string to the console
        console.log("Message to send: " + messageText);

        // Create an SNS object
        var sns = new AWS.SNS();

        // Populate the parameters for the publish operation
        // - Message : the text of the message to send
        // - TopicArn : the ARN of the Amazon SNS topic to which you want to publish
        var params = {
            Message: messageText,
            TopicArn: "arn:aws:sns:us-east-1:123456789012:MyIoTButtonSNSTopic"
         };
         sns.publish(params, context.done);
    };
```

Note
Replace the value of `TopicArn` with the ARN of the Amazon SNS topic you created previously.

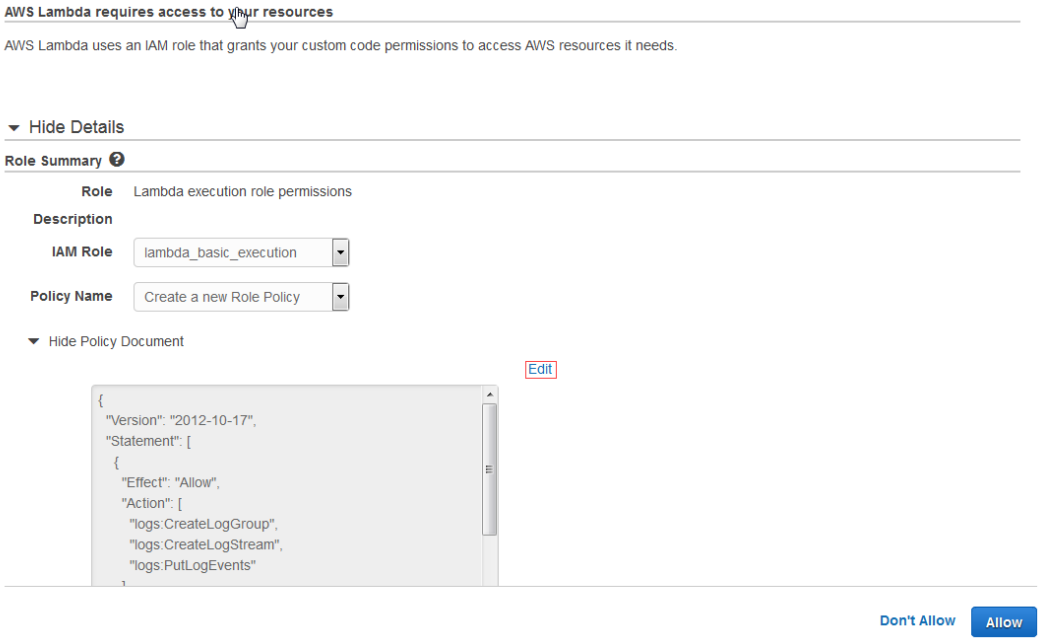
9. Scroll down to the **Lambda function handler and role** section of the page. For **Role**, choose **Create a custom role**. The IAM console opens, allowing you to create an IAM role that Lambda can assume when executing the Lambda function.

   To edit the role's policy to give it permission to publish to your Amazon SNS topic:

   a. Choose **View Policy Document**.



   b. Choose **Edit** to edit the role's policy.



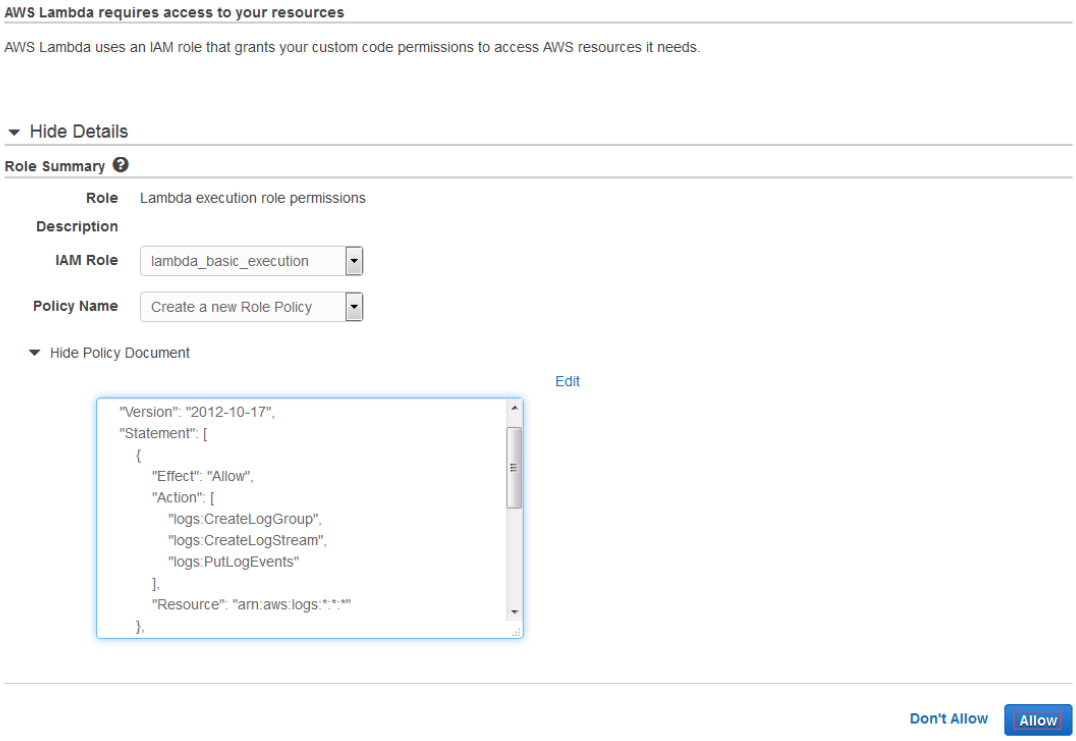
c. Replace the policy document with the following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-east-1:123456789012:MyIoTButtonSNSTopic"
        }
    ]
}
```

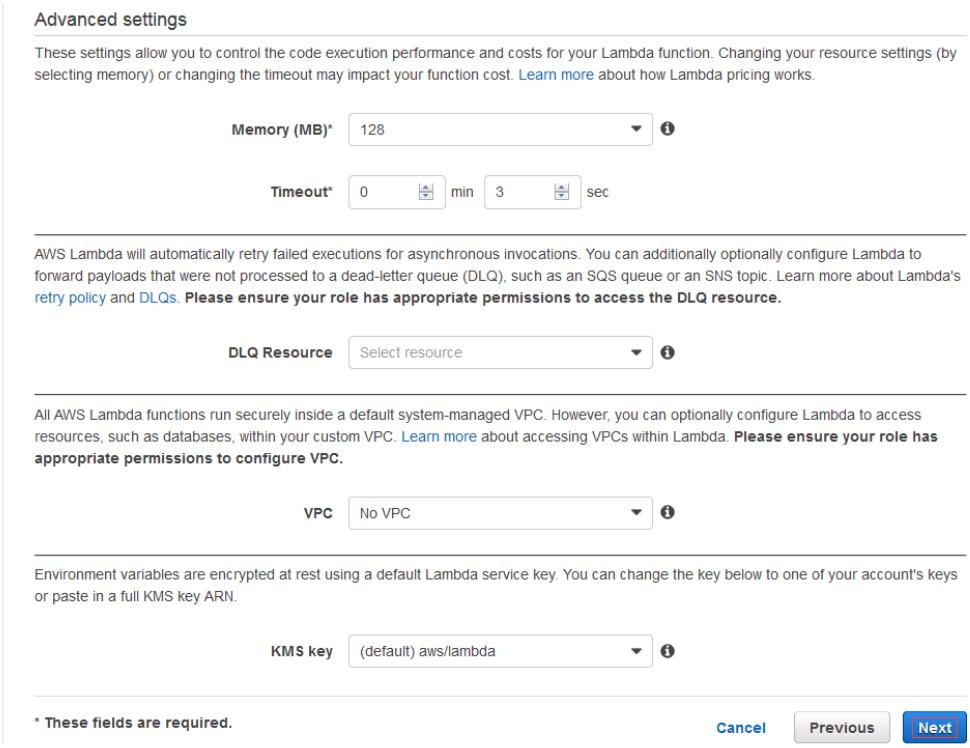
This policy document adds permission to publish to your Amazon SNS topic.

Note
Replace the value of the second `Resource` with the ARN of the Amazon SNS topic you created previously.

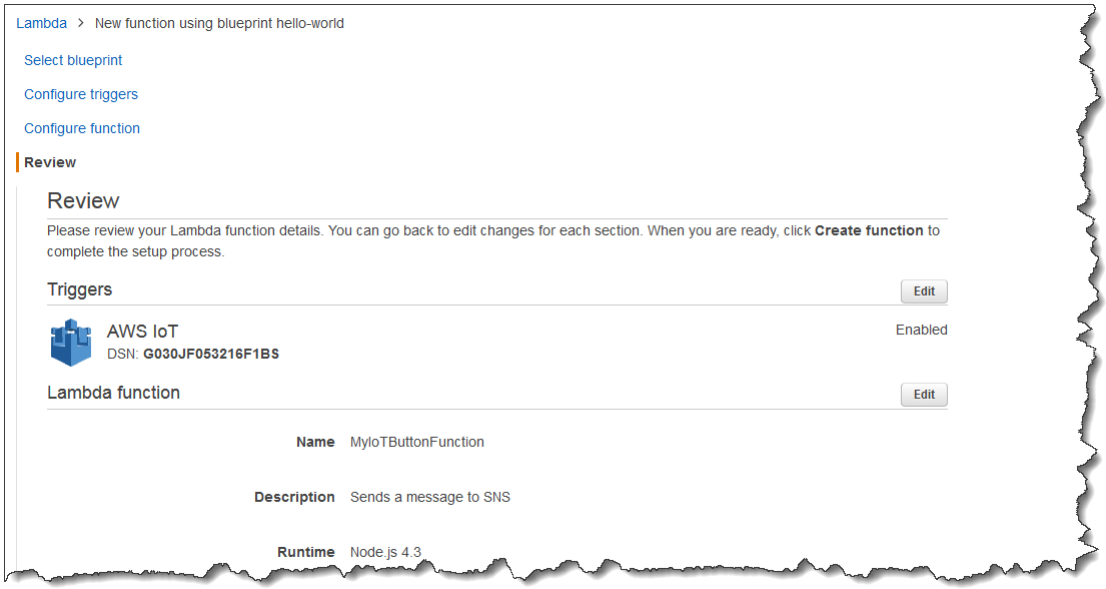
10. Choose **Allow**.



11. Leave the settings on the **Advanced settings** page at their defaults, and choose **Next**.



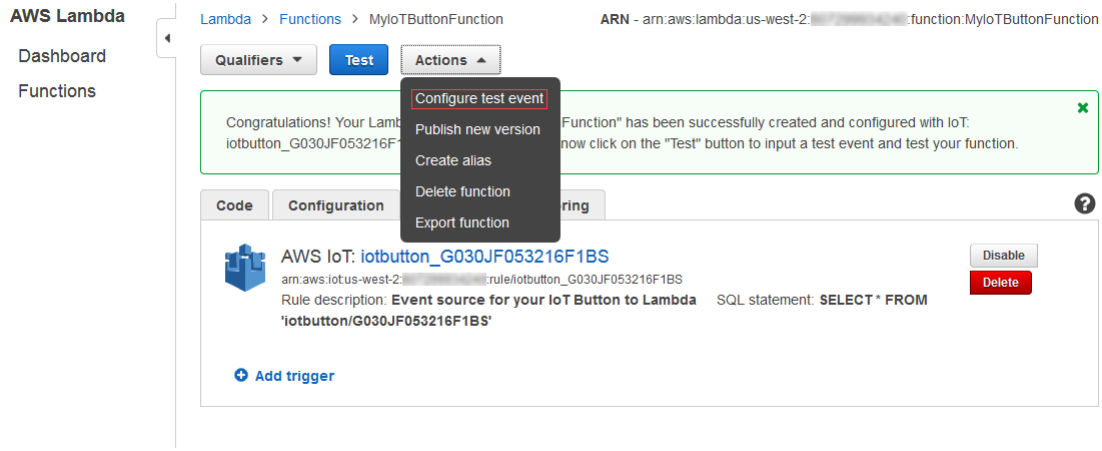
12. On the **Review** page, choose **Create function**.



Test Your Lambda Function

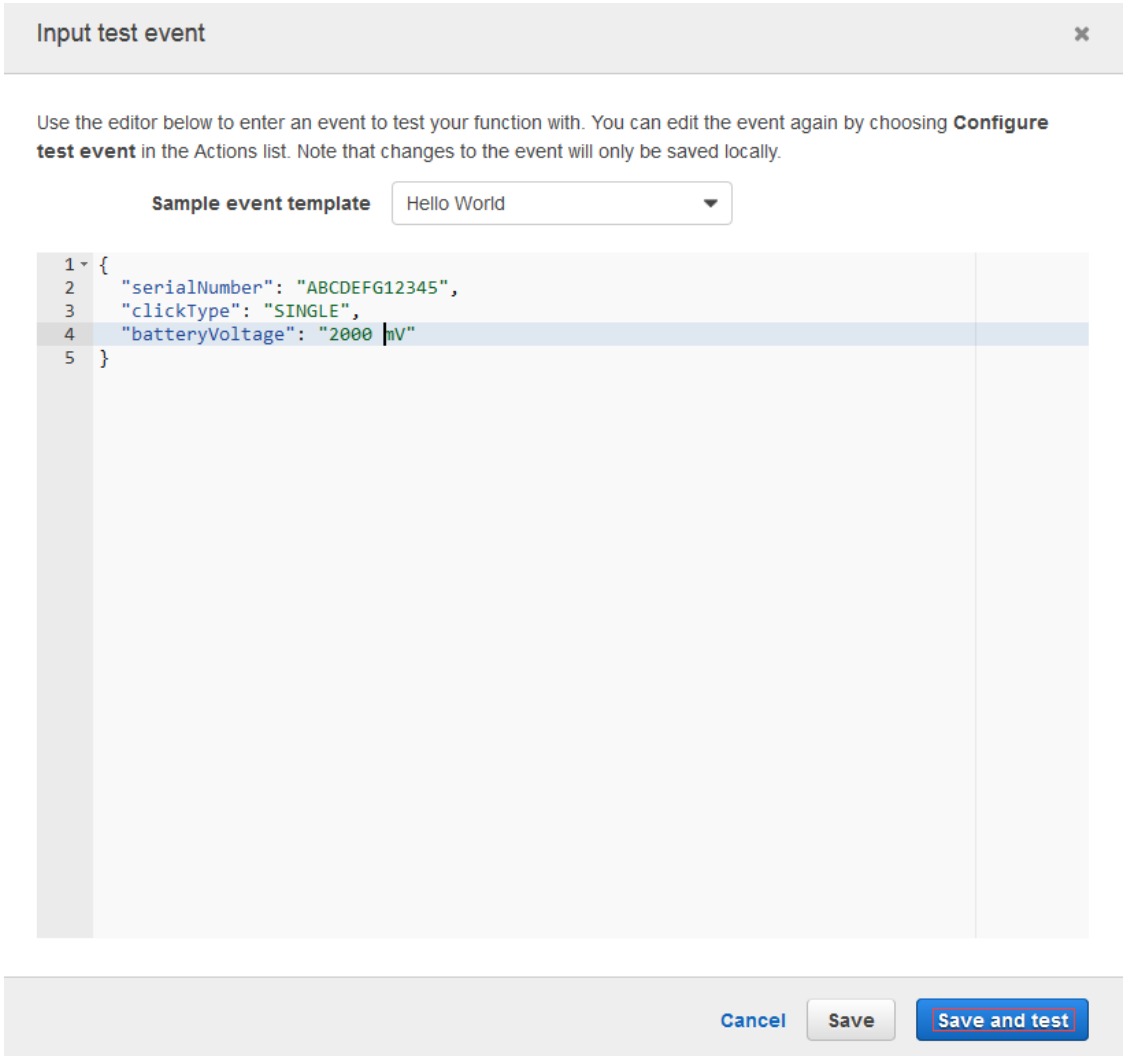
To test the Lambda function:

1. From the **Actions** menu, choose **Configure test event**.

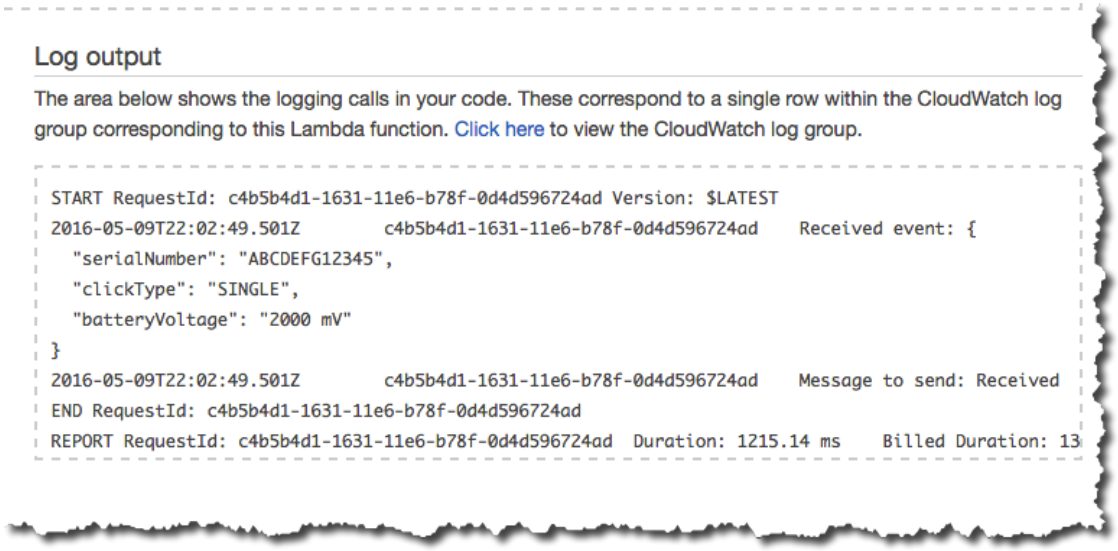


2. Copy and paste the following JSON into the **Input test event** page, and then choose **Save and test**.

```
{
    "serialNumber": "ABCDEFG12345",
    "clickType": "SINGLE",
    "batteryVoltage": "2000 mV"
}
```



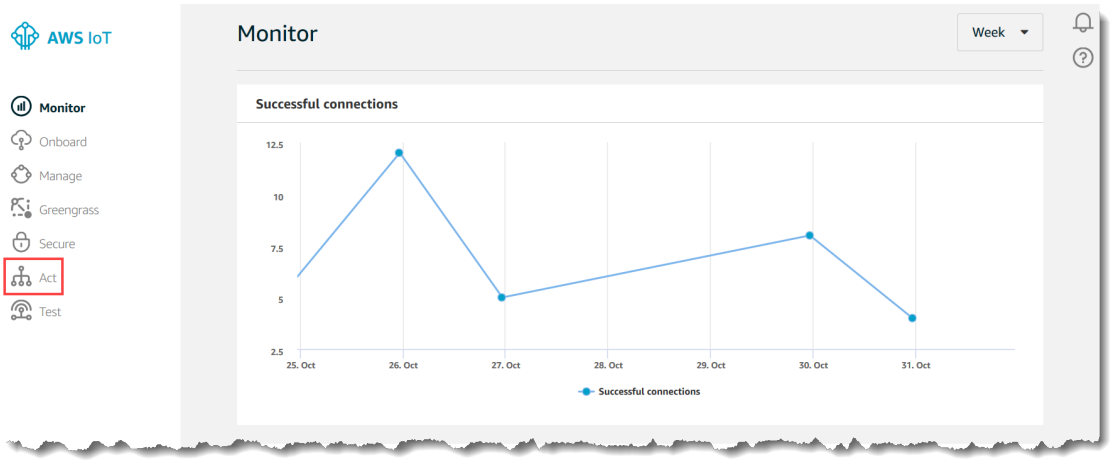
3. In the AWS Lambda console, scroll to the bottom of the page. The **Log output** section displays the output the Lambda function has written to the console.



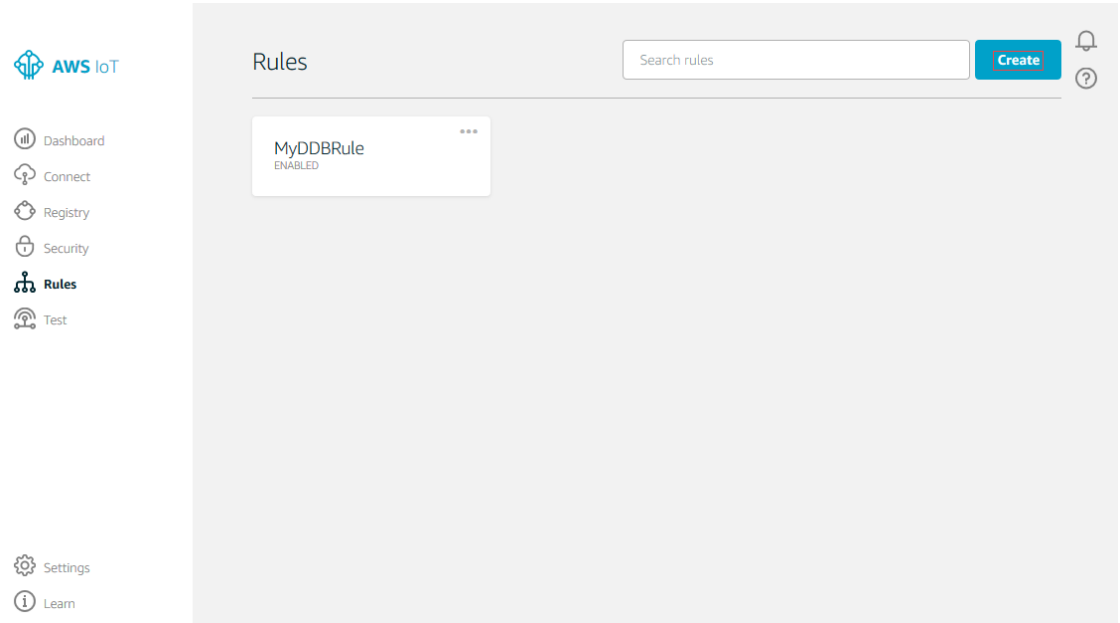
Creating a Lambda Rule

Now that you have created a Lambda function, you can create a rule that invokes the Lambda function.

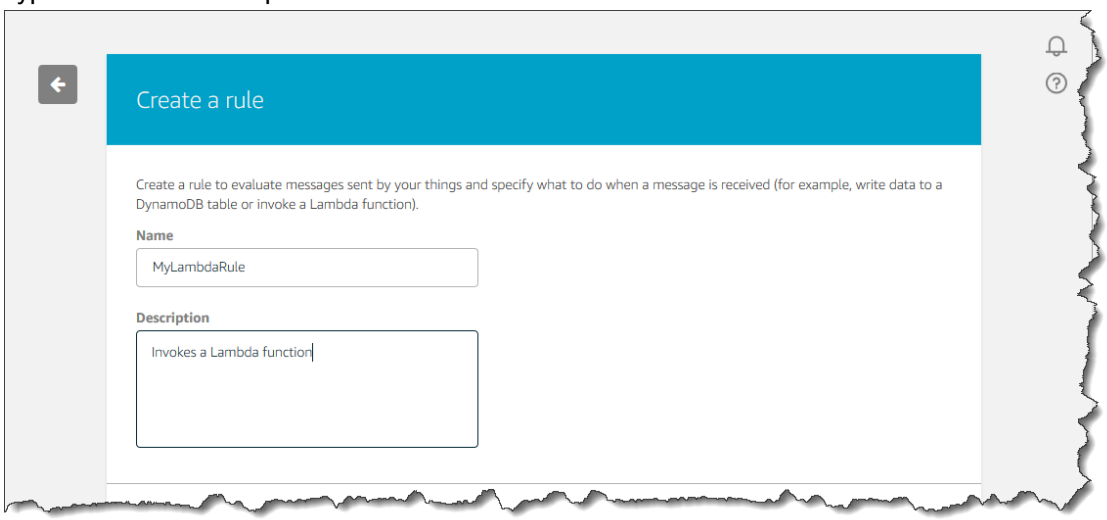
1. In the AWS IoT console, in the left navigation pane, choose **Rules**.



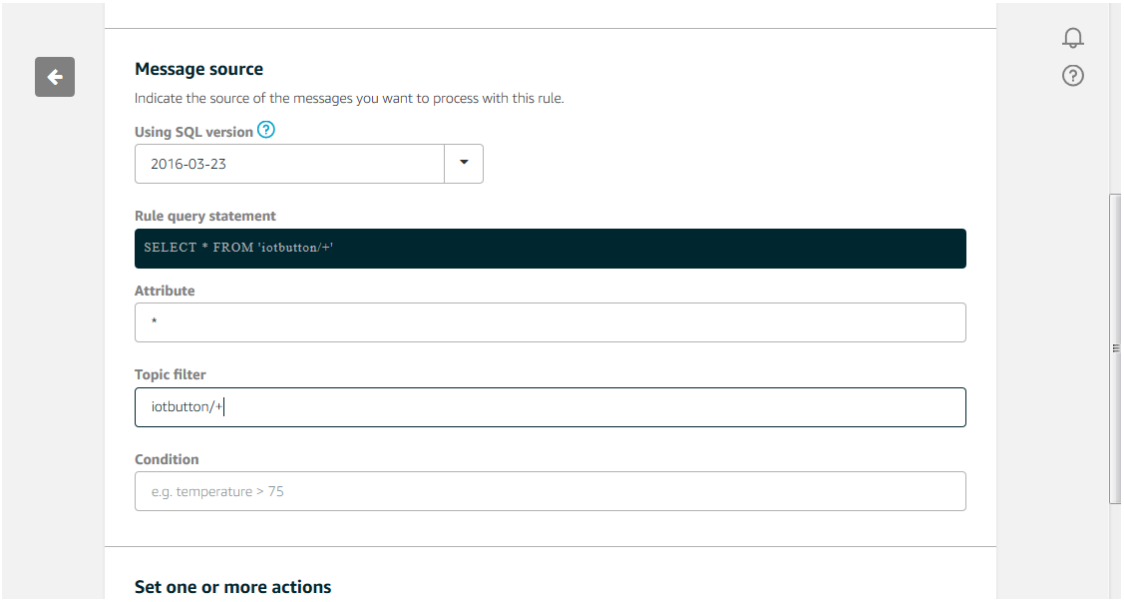
2. On the **Rules** page, choose **Create**.



3. Type a name and description for the rule.

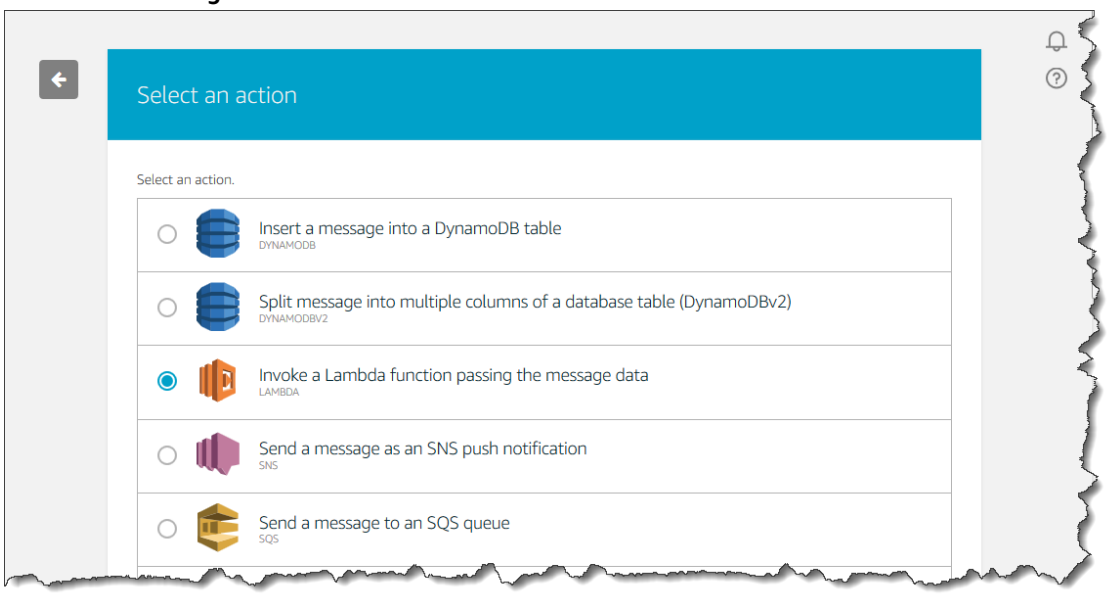


4. Enter the following settings for the rule:

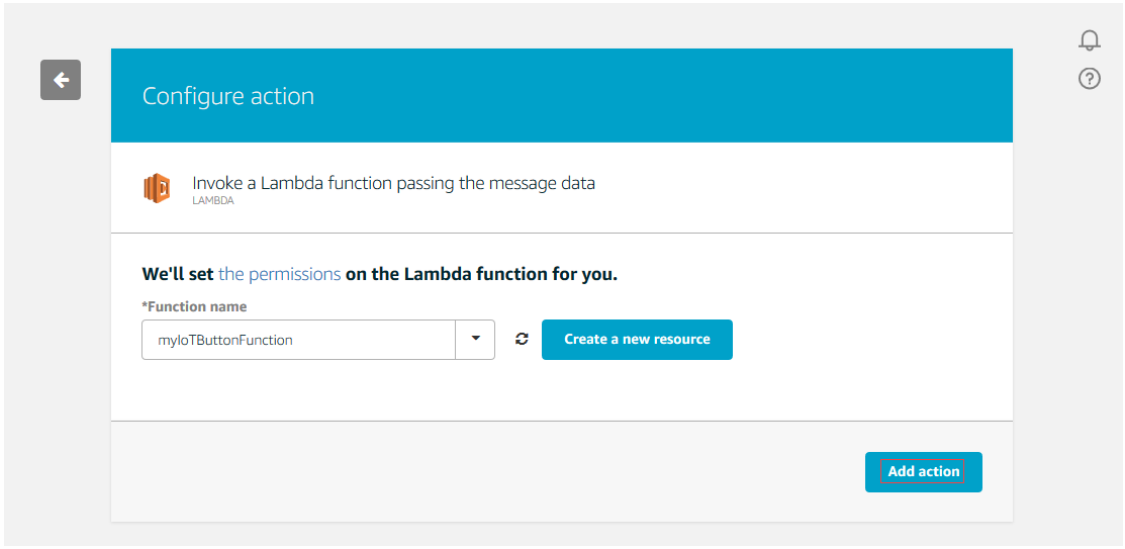


5. In **Set one or more actions**, choose **Add action**.

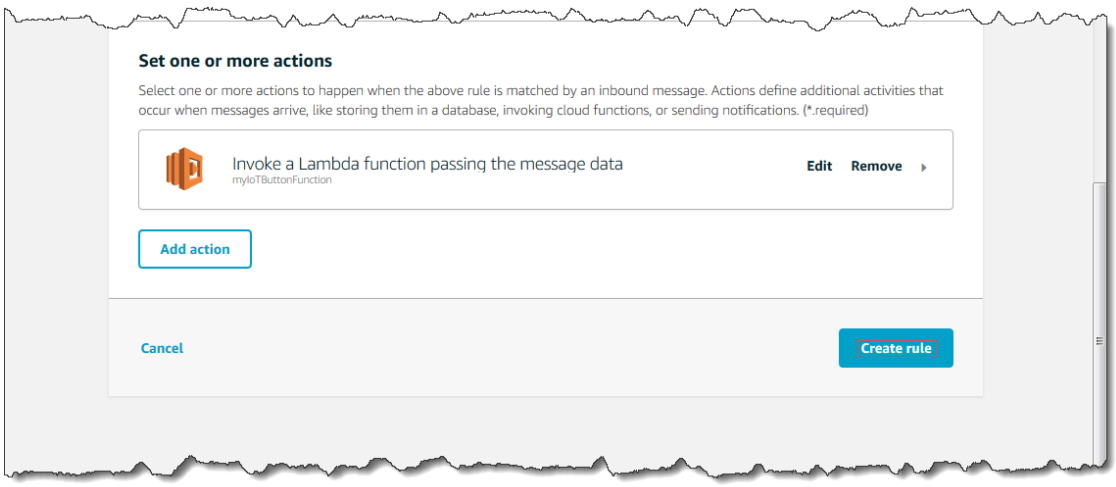
6. On the **Select an action** page, select **Invoke a Lambda function passing the message data**, and then choose **Configure action**.



7. From the **Function name** drop-down list, choose your Lambda function name, then choose **Add action**.



8. Choose **Create rule** to create your Lambda function.



Test Your Lambda Rule

In this tutorial, we assume you have completed the AWS IoT Getting Started Tutorial (p. 5), which covers:

- Configuring an AWS IoT button.
- Creating and subscribing to an Amazon SNS topic with a cell phone number.

Now that your button is configured and connected to Wi-Fi and you have configured an Amazon SNS topic, you can press the button to test your Lambda rule. You should receive an SMS text message on your phone that contains:

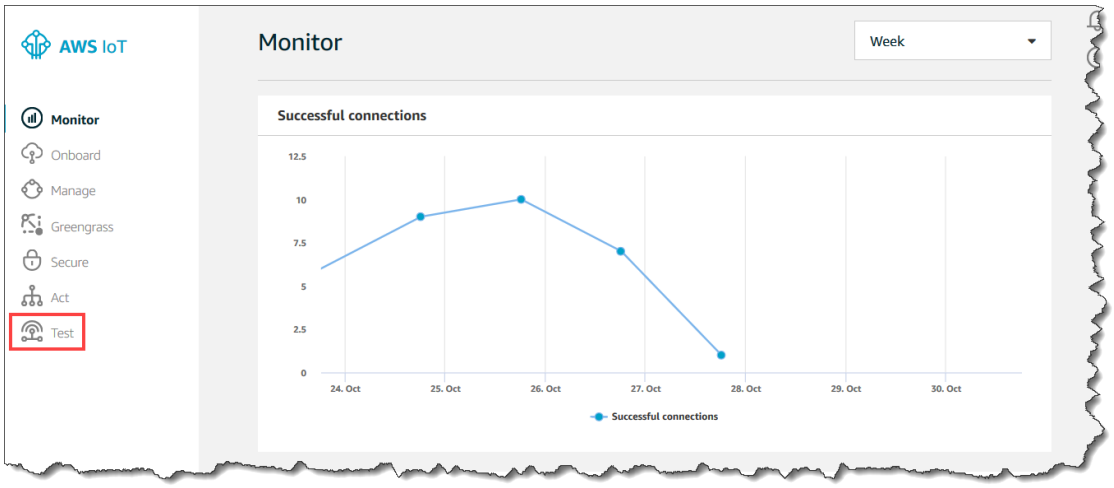
- The serial number of your button.
- The type of button press (SINGLE or DOUBLE).
- The battery voltage.

The message should look like the following:

```
IOT BUTTON> {
    "serialNumber" : "ABCDEFG12345",
    "clickType" : "SINGLE",
    "batteryVoltage" : "2000 mV"
}
```

If you do not have a button, you can buy one here or you can use the AWS IoT MQTT client instead.

1. In the AWS IoT console, choose **Test**.



2. On the **MQTT client** page, in the **Publish** section, in **Specify a topic**, type **`iotbutton/ABCDEFG12345`**.

   In **Payload**, type the following JSON, and then choose **Publish to topic**.

```
{
    "serialNumber" : "ABCDEFG12345",
    "clickType" : "SINGLE",
    "batteryVoltage" : "2000 mV"
}
```

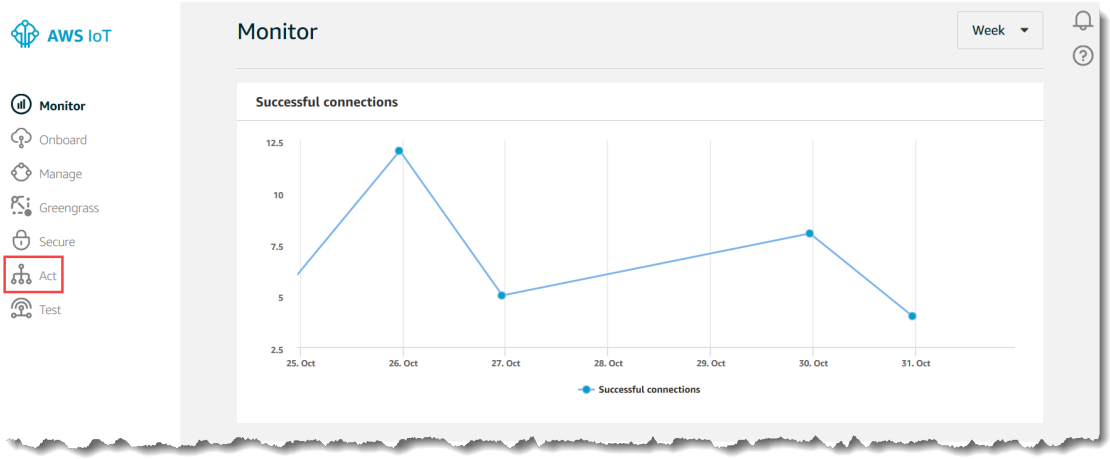
3. You should receive a message on your cell phone.

Creating an Amazon SNS Rule

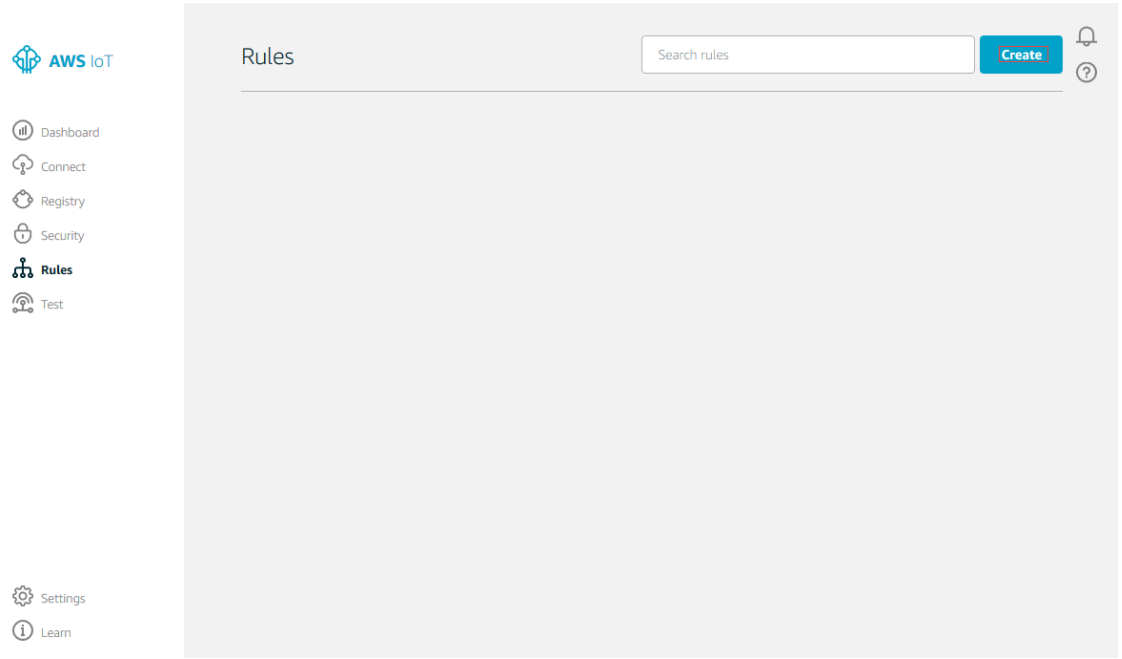
You can define a rule that sends message data to an Amazon SNS topic. In this tutorial, you will create a rule that sends the name of the AWS IoT thing that triggered the rule to all subscribers of an Amazon SNS topic.

To create a rule with an SNS action:

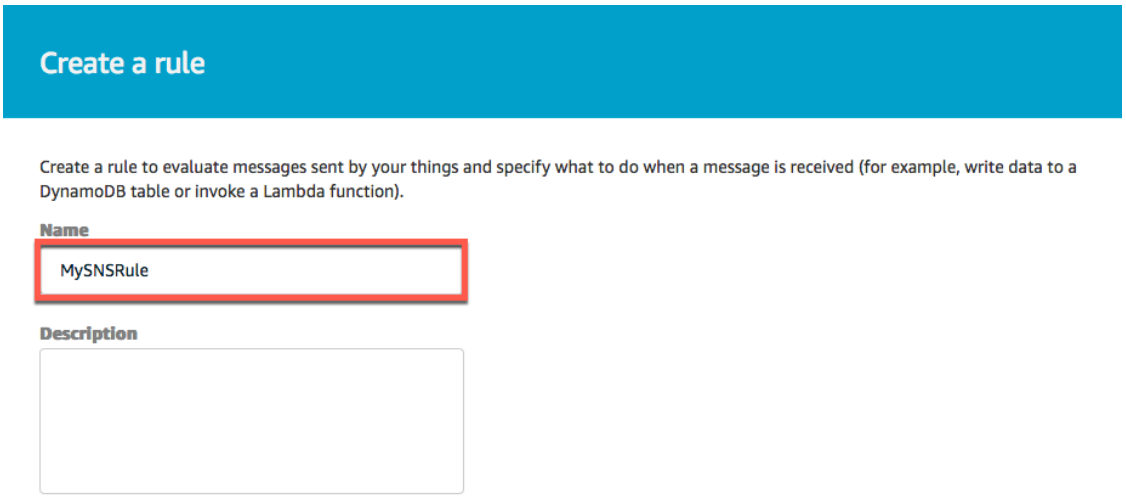
1. In the AWS IoT console, in the left navigation pane, choose **Rules**.



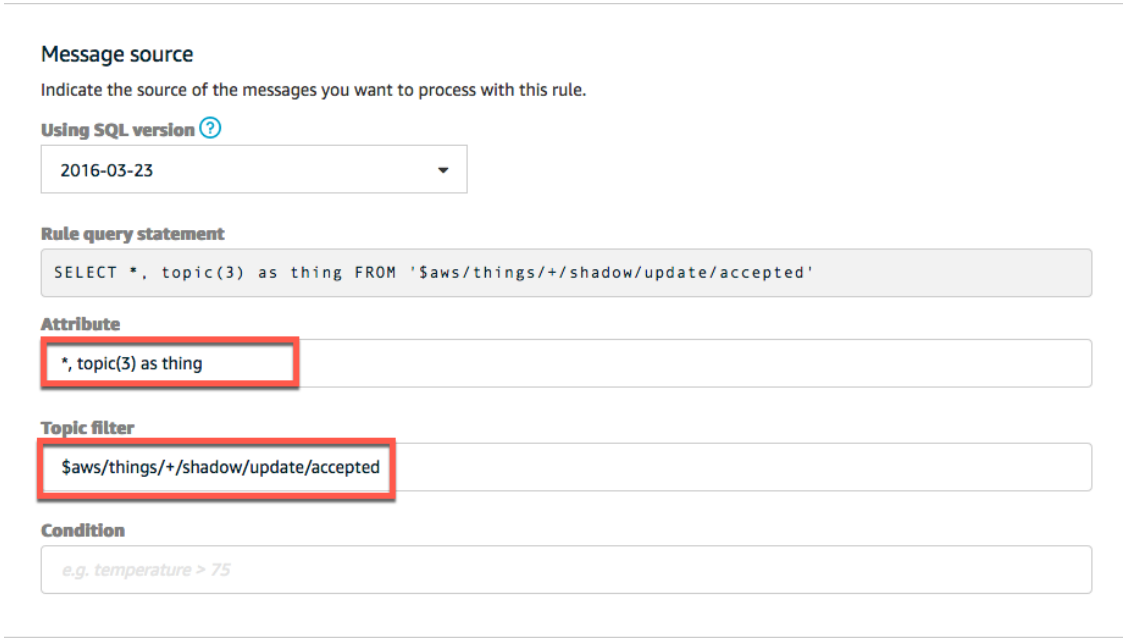
2. On the **Rules** page, choose **Create**.



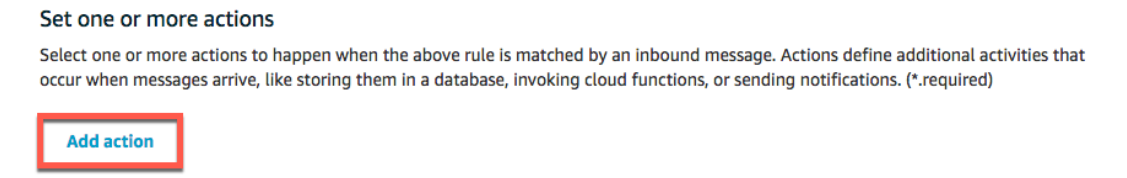
3. Type a name for your rule.



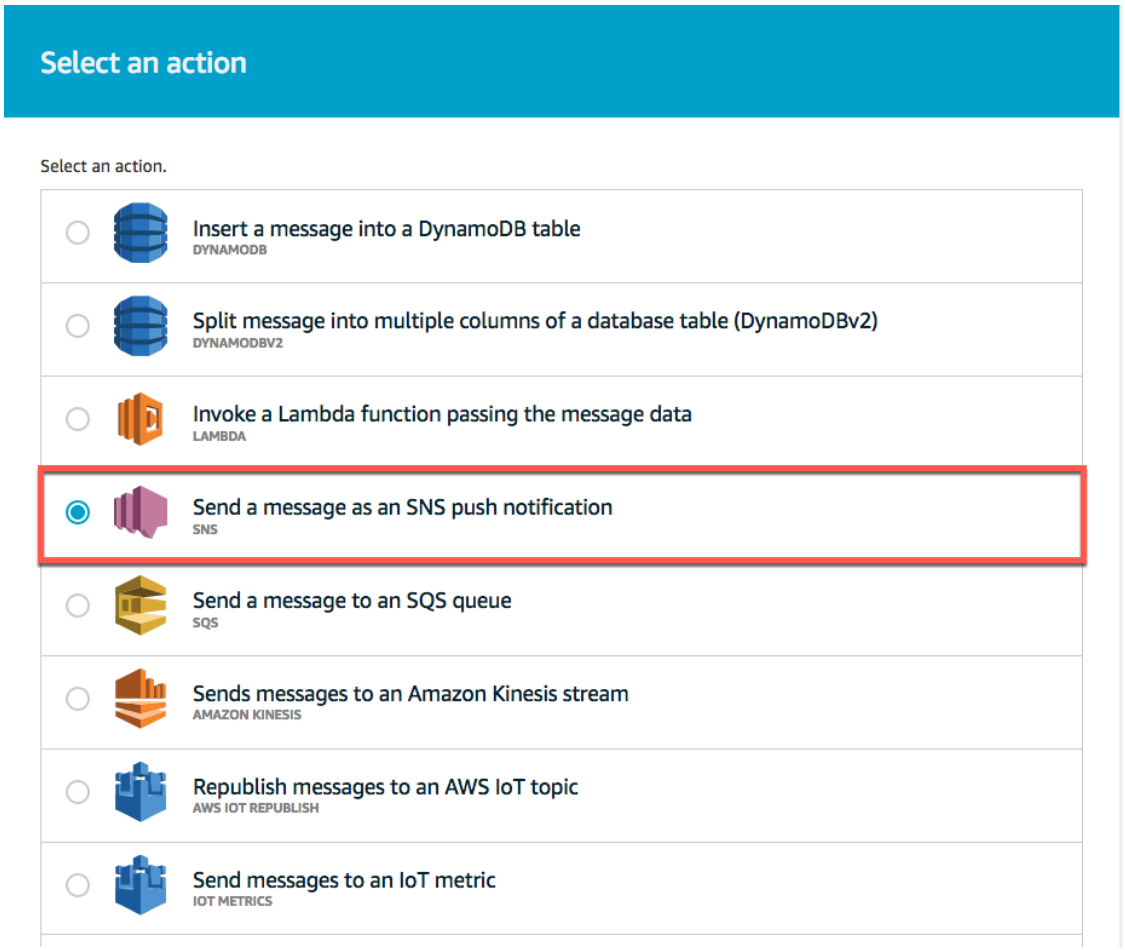
4. Under **Message source**, for **Attribute** type `*, topic(3)`. For **Topic filter**, type `$aws/things/+/shadow/update/accepted`. The topic filter specifies the topics that, when a message is published to them, trigger the rule's action. The + used in the topic filter is a wildcard character that matches any thing name. The attribute appends the thing name onto the message contents.



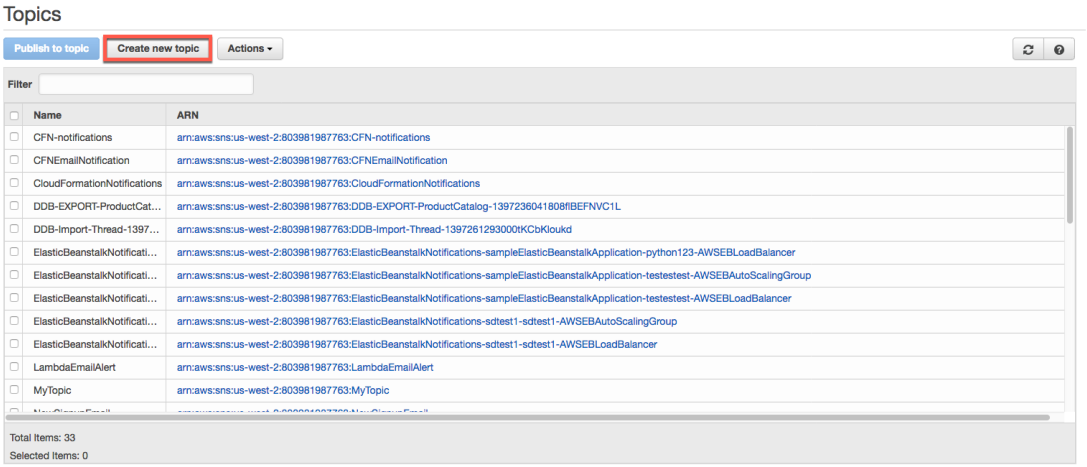
5. In the **Set one or more actions** section, choose **Add action**.



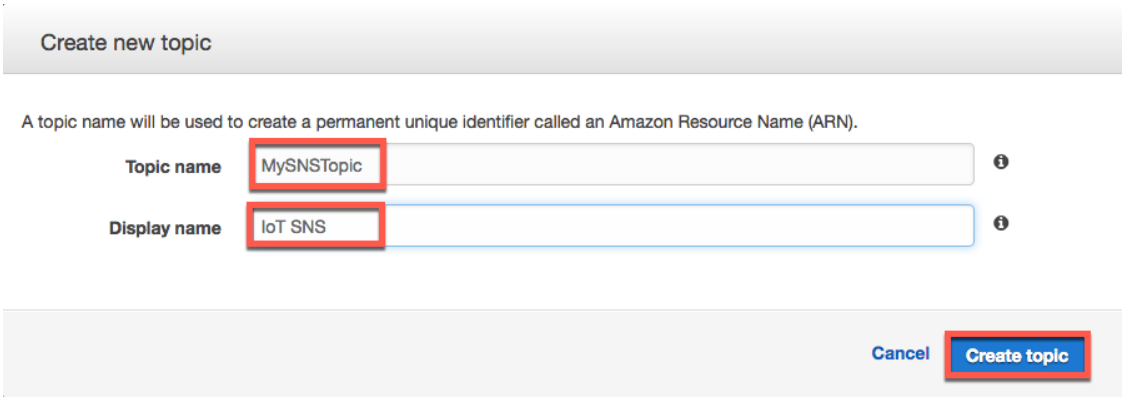
6. Under **Select an action**, select **Send a message as an SNS push notification**, and then choose **Configure action**. (This button is not shown in screenshot).



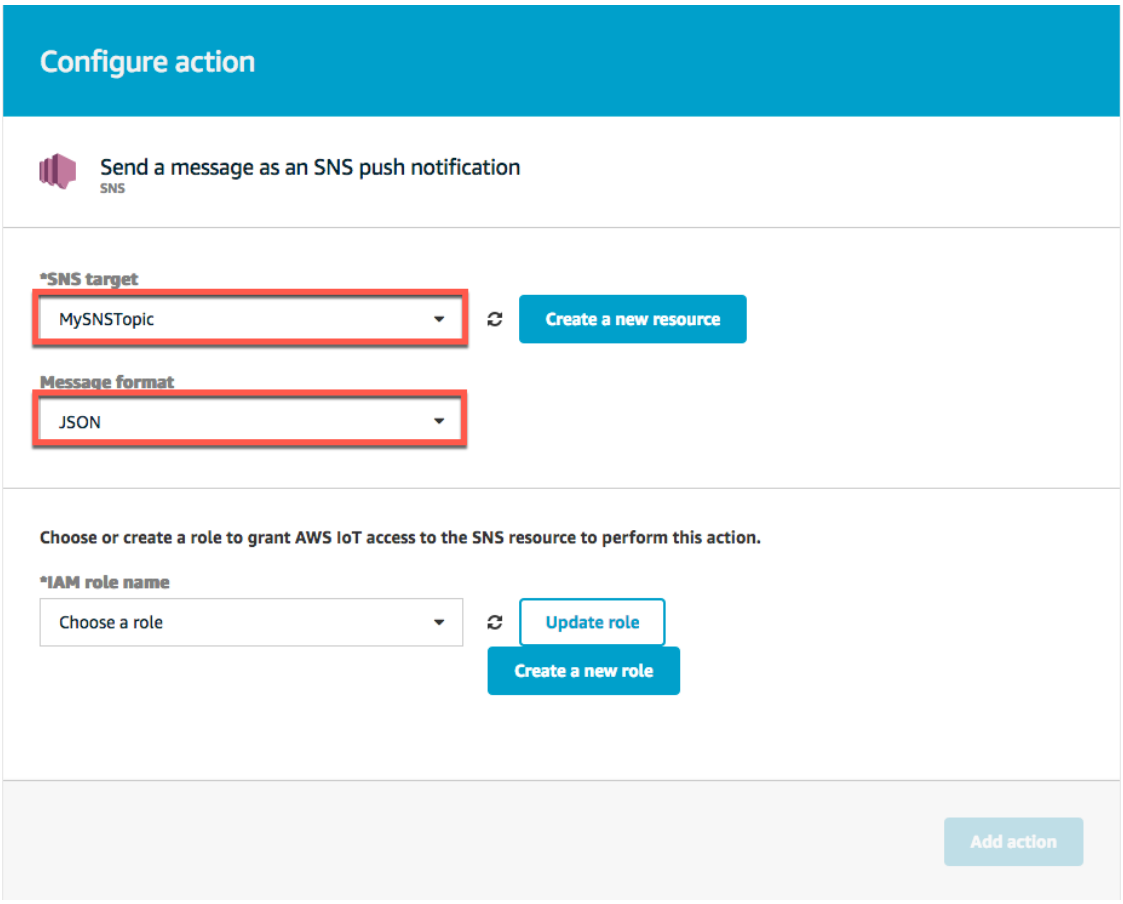
7. Choose **Create new topic**.



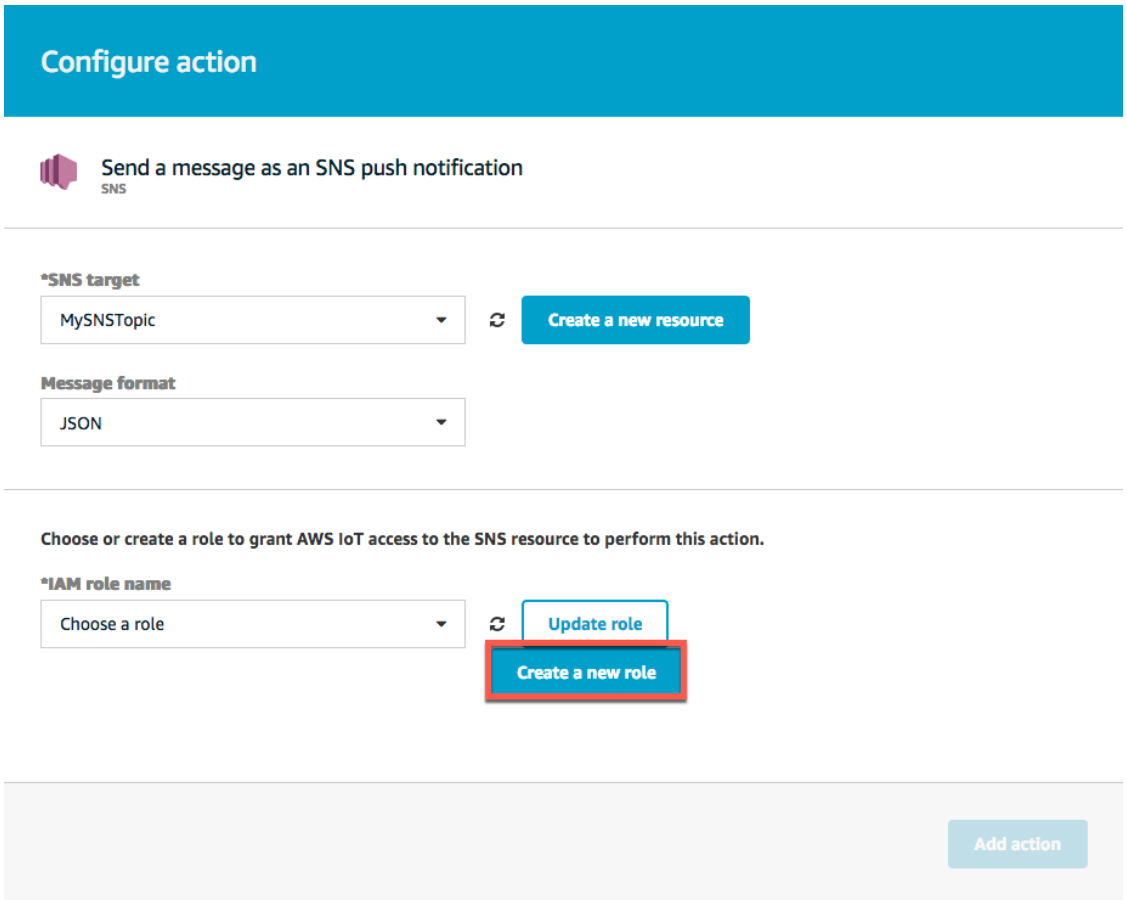
8. A new tab opens in your browser. Type a name and description for your SNS topic, and then choose **Create topic**.



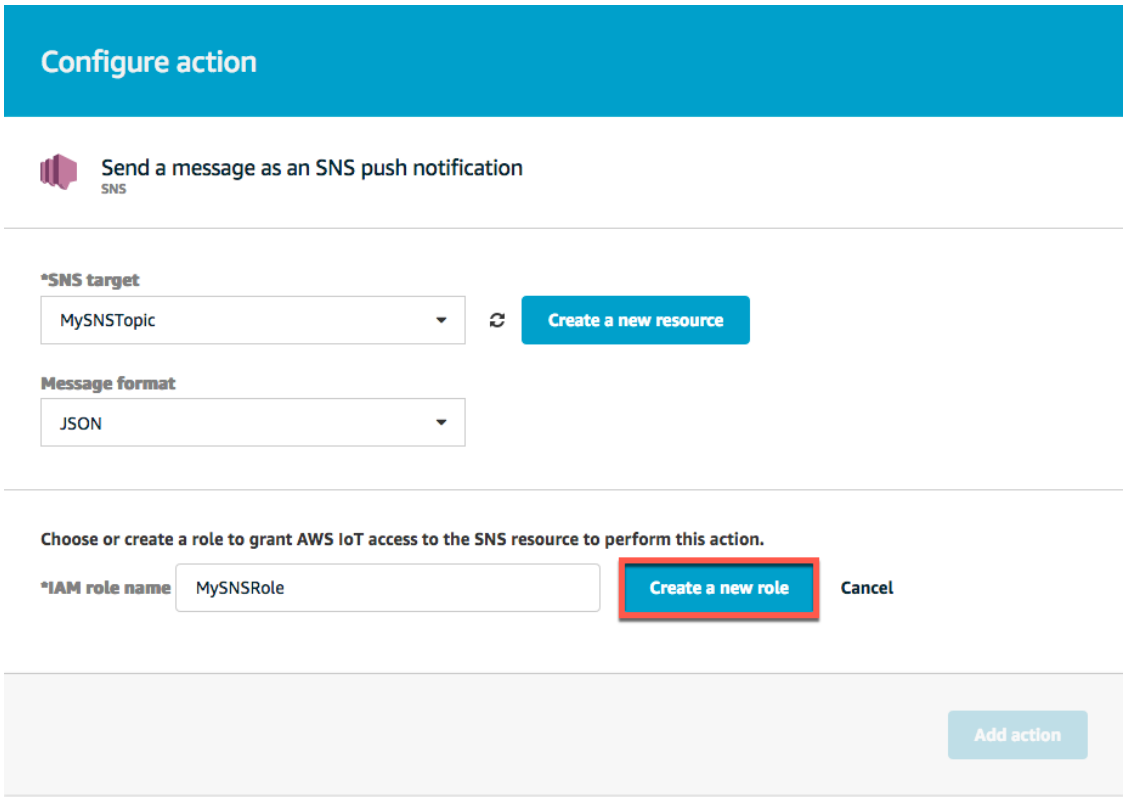
9. Switch to the browser tab where the AWS IoT console is open. For **SNS target**, choose the SNS topic you just created. For **Message format**, choose **JSON**.



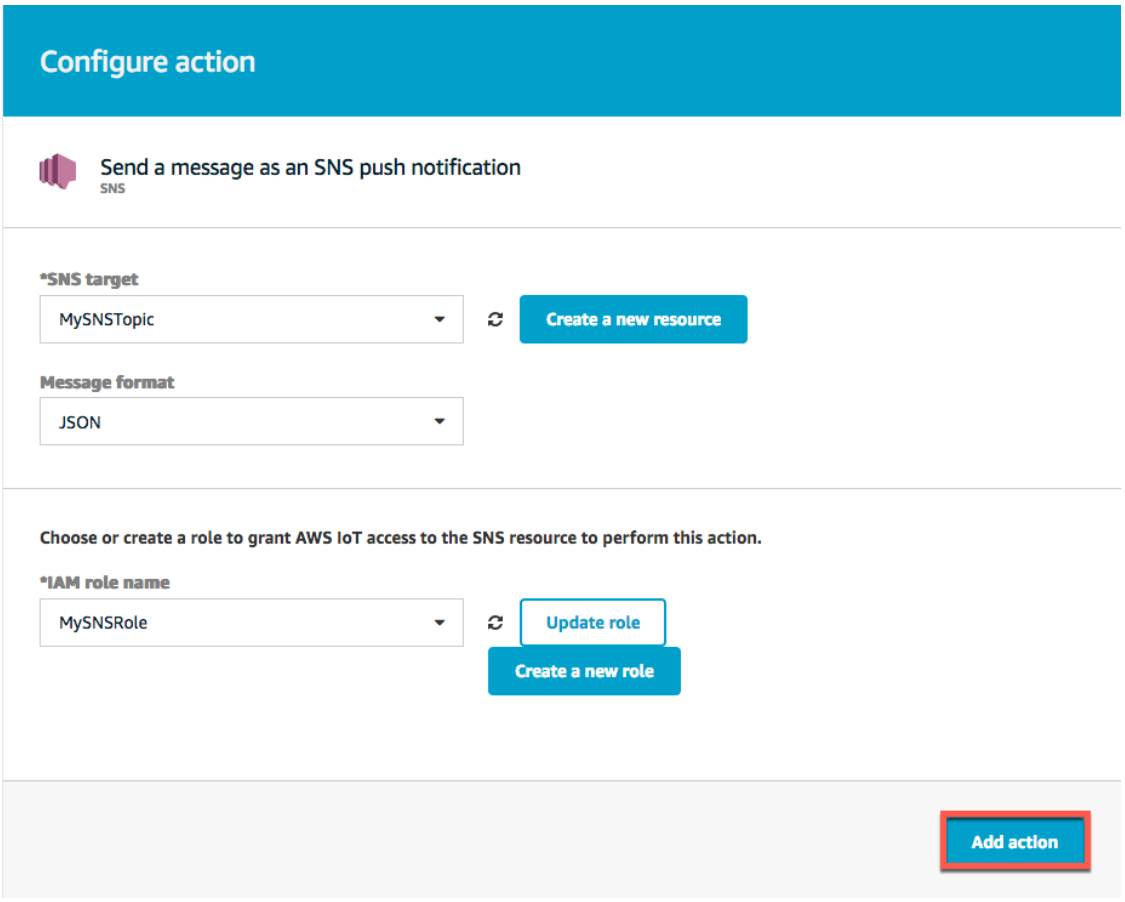
10. For **IAM role name**, choose **Create a new role**.



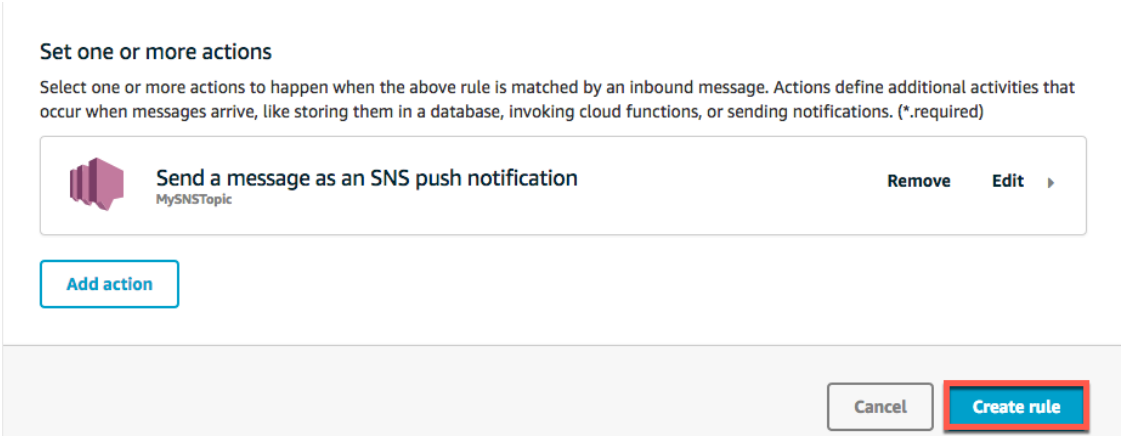
11. Type a name for the role, and then choose **Create a new role**.



12. Select the role you just created, and then choose **Add action**.



13. Choose **Create rule**.



You have now created the rule. To test the rule, add a subscription to the SNS topic you created, and update the thing shadow of any AWS IoT thing. You can use the AWS IoT console to find a thing, open its detail page, and change the thing's shadow. When the Thing Shadow service is notified of the change, it will publish a message on `$aws/things/MySNSThing/shadow/update/accepted`. Your rule is triggered and all subscribers to your SNS topic receive a message that contains your thing's name.

AWS IoT SDK Tutorials

The AWS IoT Device SDKs help you to easily and quickly connect your devices to AWS IoT. The AWS IoT Device SDKs include open-source libraries, developer guides with samples, and porting guides so that you can build innovative IoT products or solutions on your choice of hardware platforms.

This guide provides step-by-step instructions for connecting your Raspberry Pi to the AWS IoT platform and setting it up for use with the AWS IoT Embedded C SDK and Device SDK for Javascript. After following the steps in this guide, you are able to connect to the AWS IoT platform and run sample apps included with these AWS IoT SDKs.

Contents

Connecting Your Raspberry Pi

Follow these steps to connect your Raspberry Pi to the AWS IoT platform.

Prerequisites

- A fully set up Raspberry Pi board with Internet access

  For information about setting up your Raspberry Pi, see Raspberry Pi Quickstart Guide.
- Chrome or Firefox (Iceweasel) browser

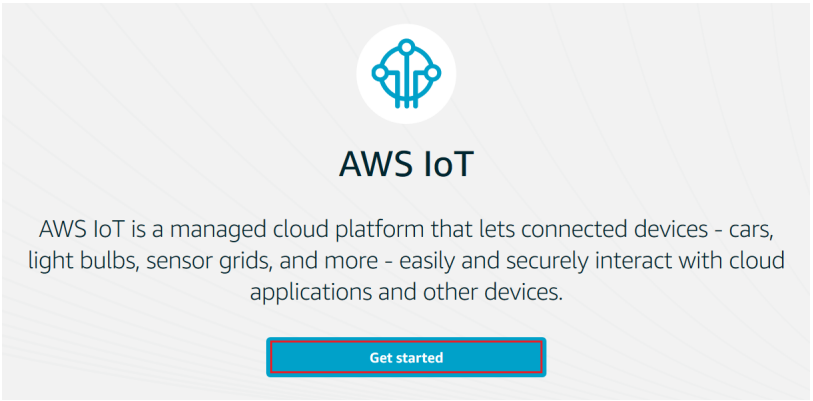
  For information about installing Iceweasel, see the instructions on the Embedded Linux wiki.

In this guide, the following hardware and software are used:

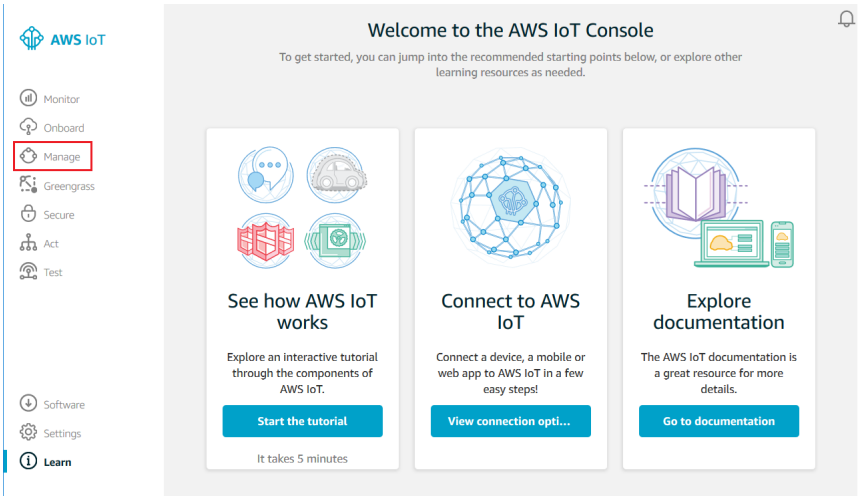
- Raspberry Pi 2 Model B
- Raspbian Wheezy
- Raspbian Jessie
- Iceweasel browser

Sign in to the AWS IoT Console

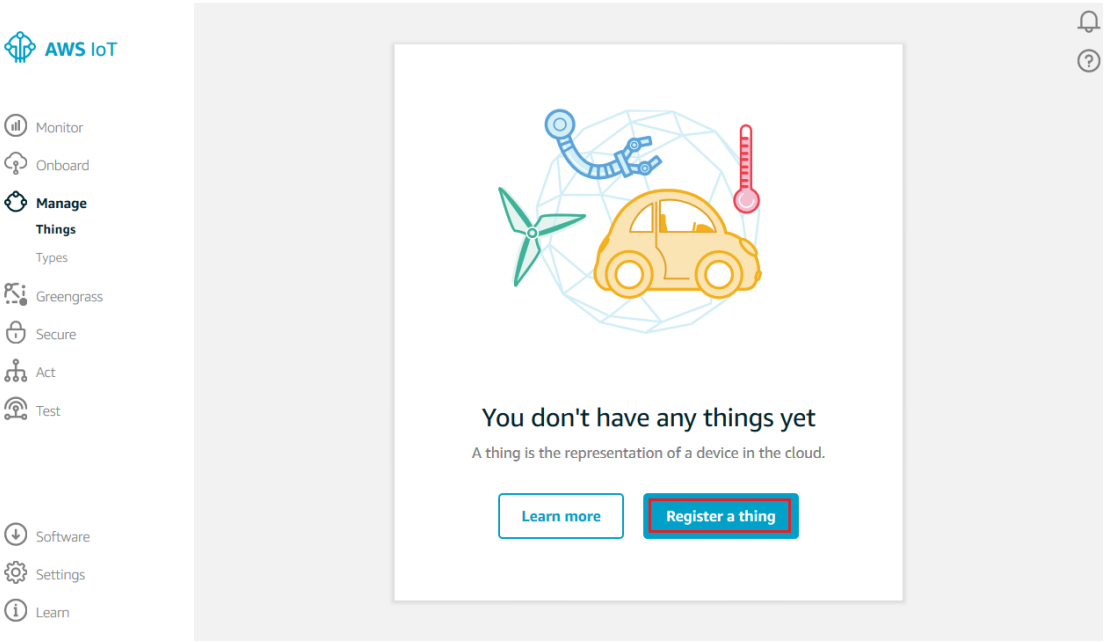
1. Turn on your Raspberry Pi and confirm you have an Internet connection.
2. Sign in to the AWS Management Console and open the AWS IoT console at https://aws.amazon.com/iot. On the **Welcome** page, choose **Get started**.



3. If this is your first time using the AWS IoT console, you see the **Welcome to the AWS IoT Console** page. In the left navigation pane, choose **Manage** to expand the choices, and then choose **Things**.



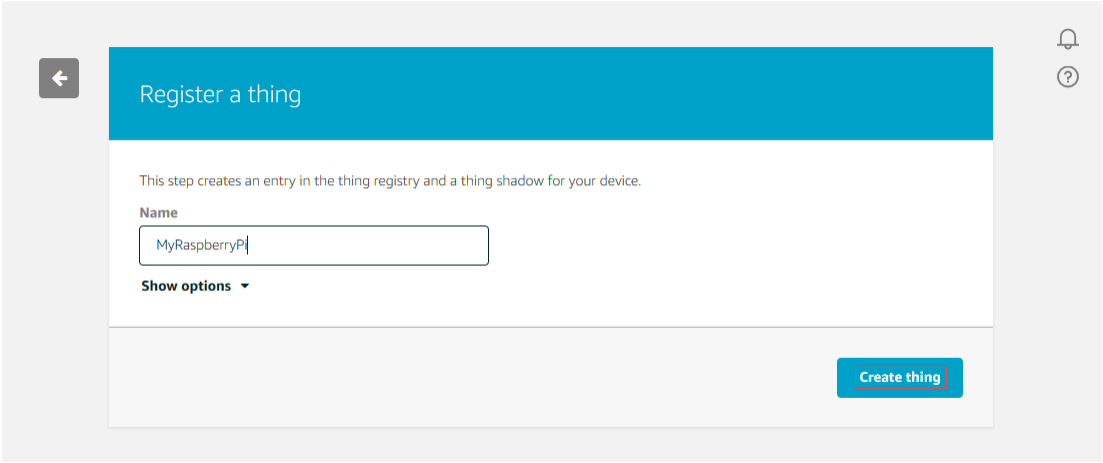
4. On the page that says **You don't have any things yet**, choose **Register a thing**. (If you have created a thing before, choose **Create**.)



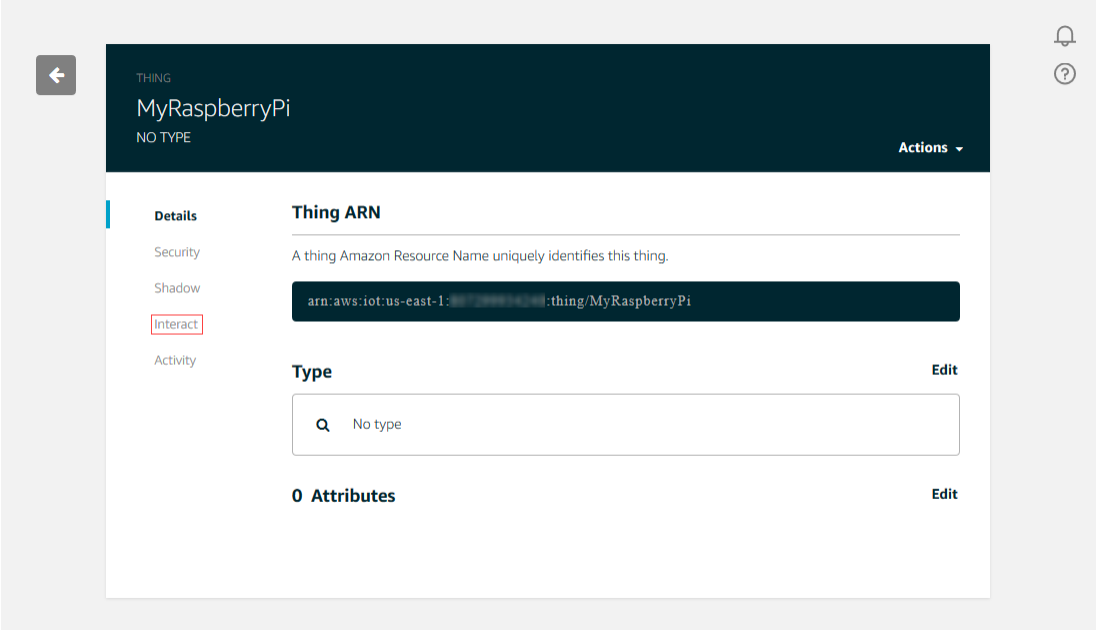
Create and Attach a Thing (Device)

A thing represents a device whose status or data is stored in the AWS Cloud. The Thing Shadows service maintains a thing shadow for each device connected to AWS IoT. Thing shadows allow you to access and modify thing state data.

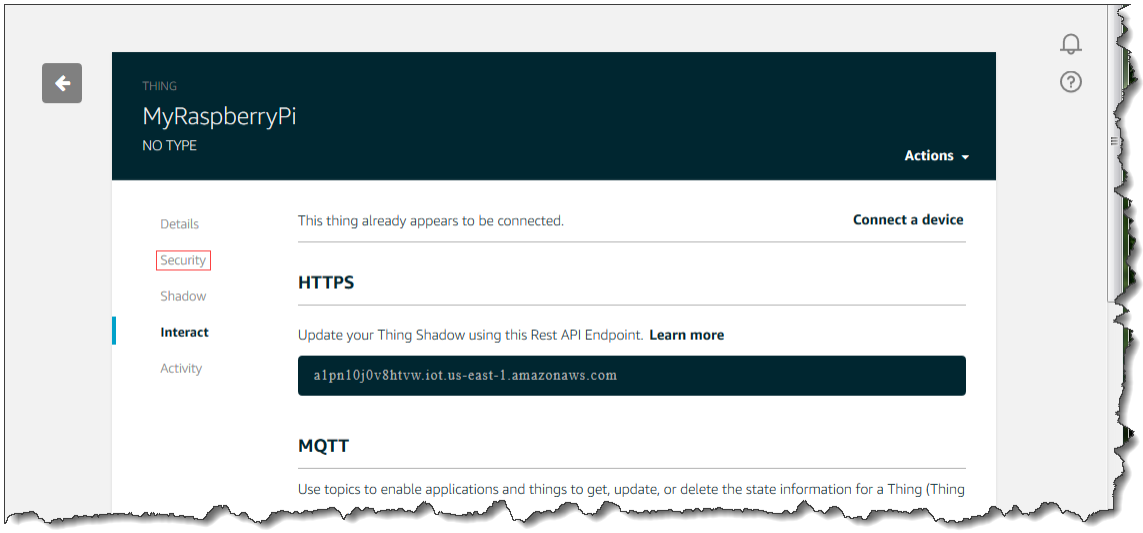
1. Type a name for the thing, and then choose **Create thing**.



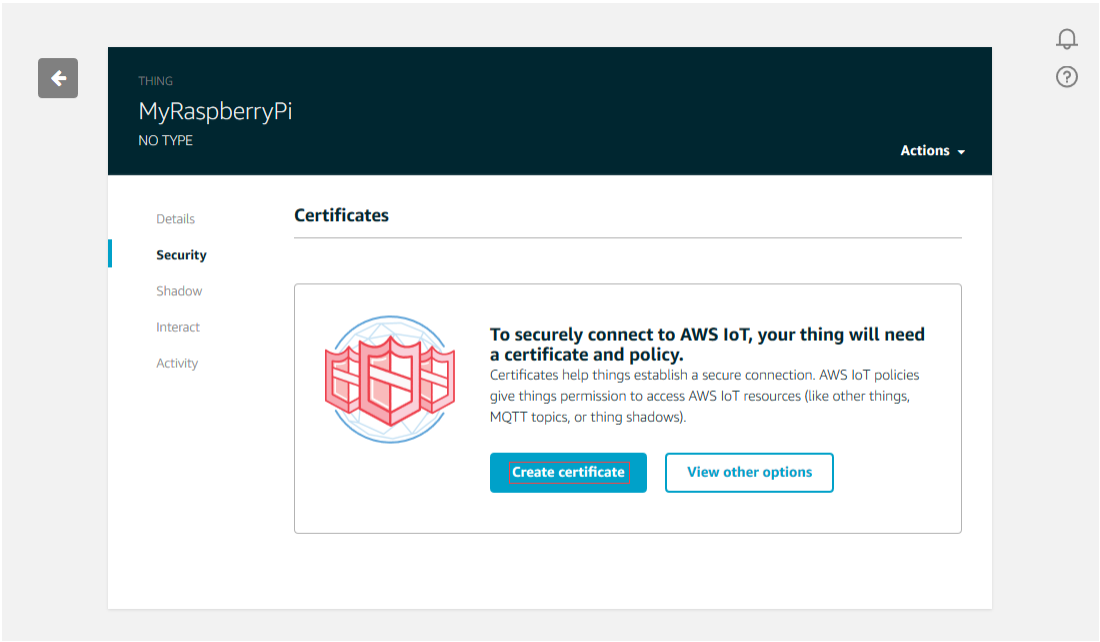
2. On the **Details** page, choose **Interact**.



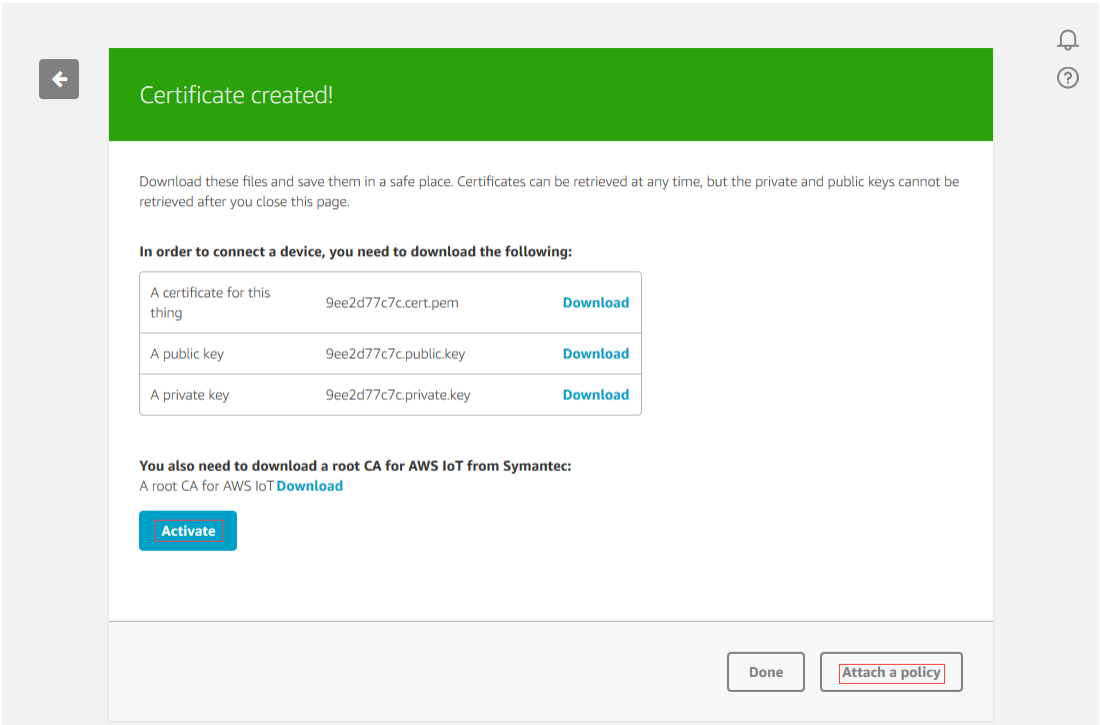
3. Make a note of the REST API endpoint. You need this value later. Choose **Security**.



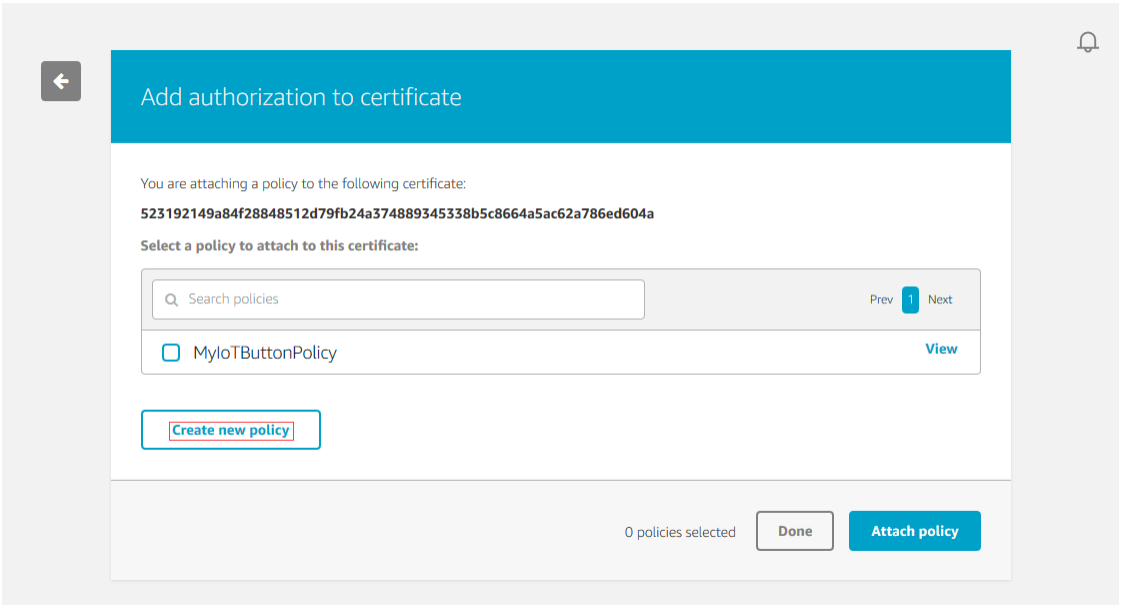
4. Choose **Create certificate**. This generates an X.509 certificate and key pair.



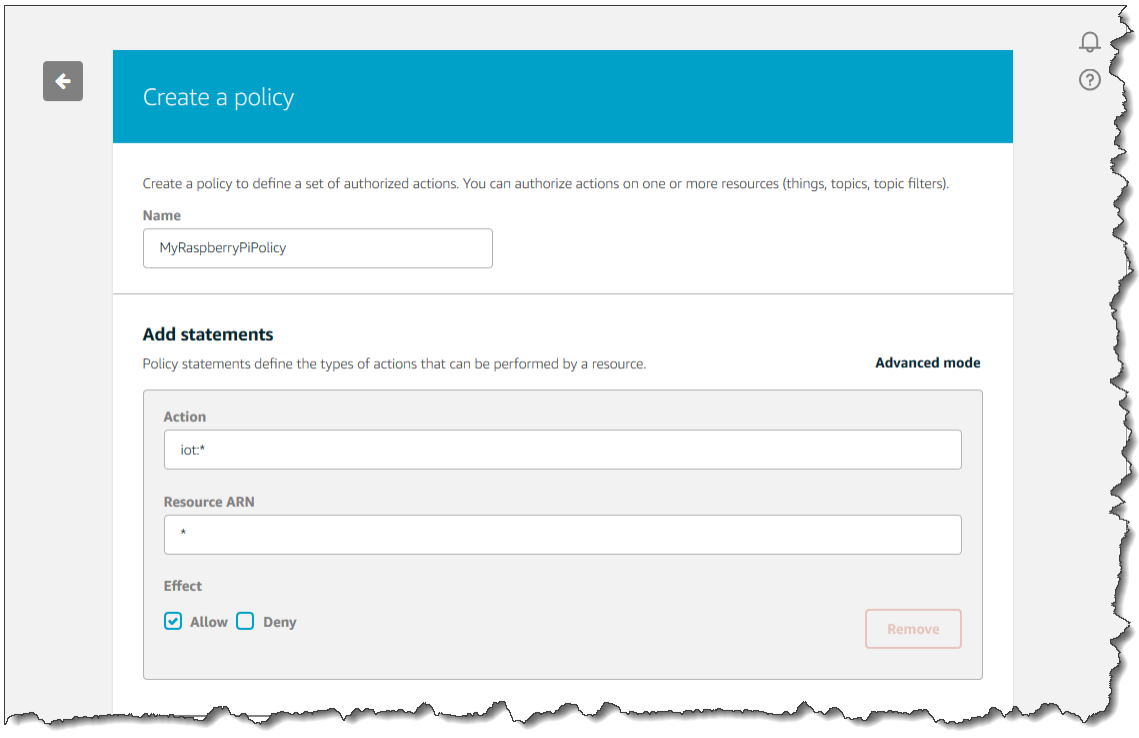
5. Create a working directory called `deviceSDK` where your files will be stored. Choose the links to download your public and private keys, certificate, and root CA and save them in the `deviceSDK` directory. Choose **Activate** to activate the X.509 certificate, then choose **Attach a policy**.



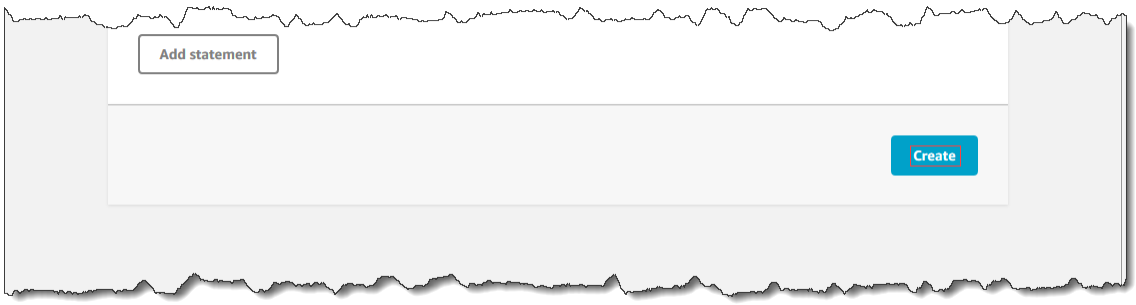
6. Choose **Create new policy**.



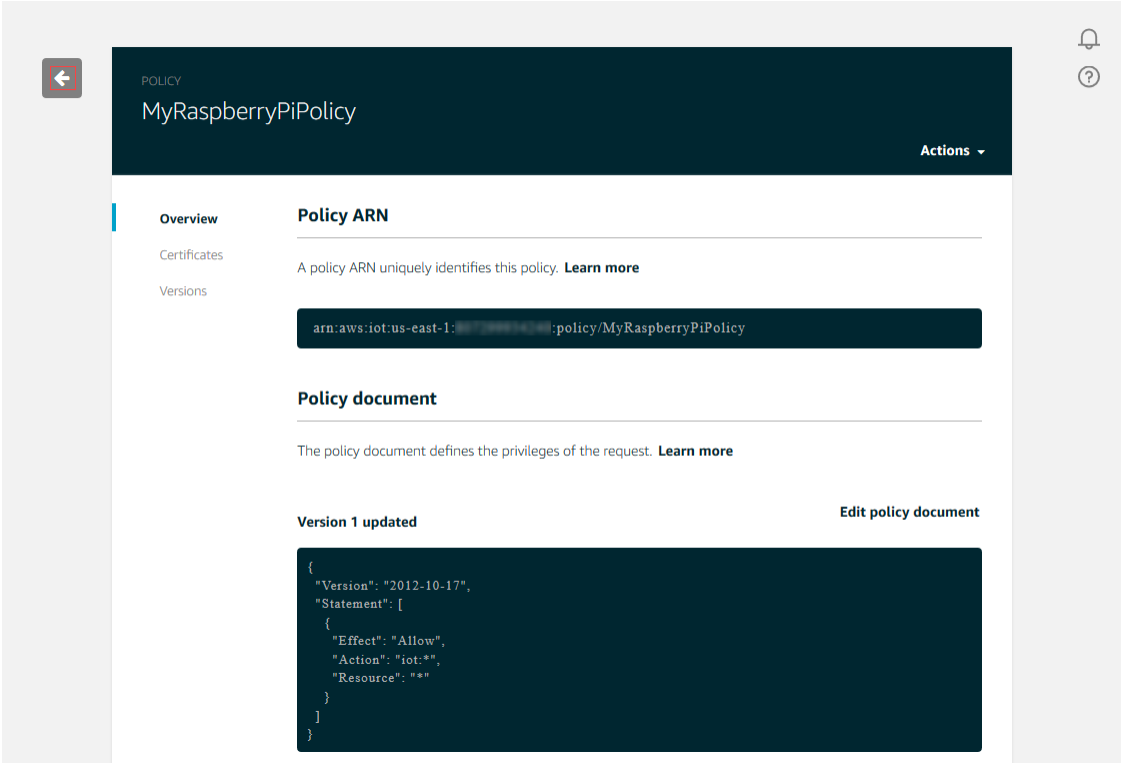
7. On the **Create a policy** page, in the **Name** field, type a name for the policy. In the **Action** field, type **`iot:*`**. In the **Resource ARN** field, type **`*`**. Select the **Allow** check box. This allows your Raspberry Pi to publish messages to AWS IoT.



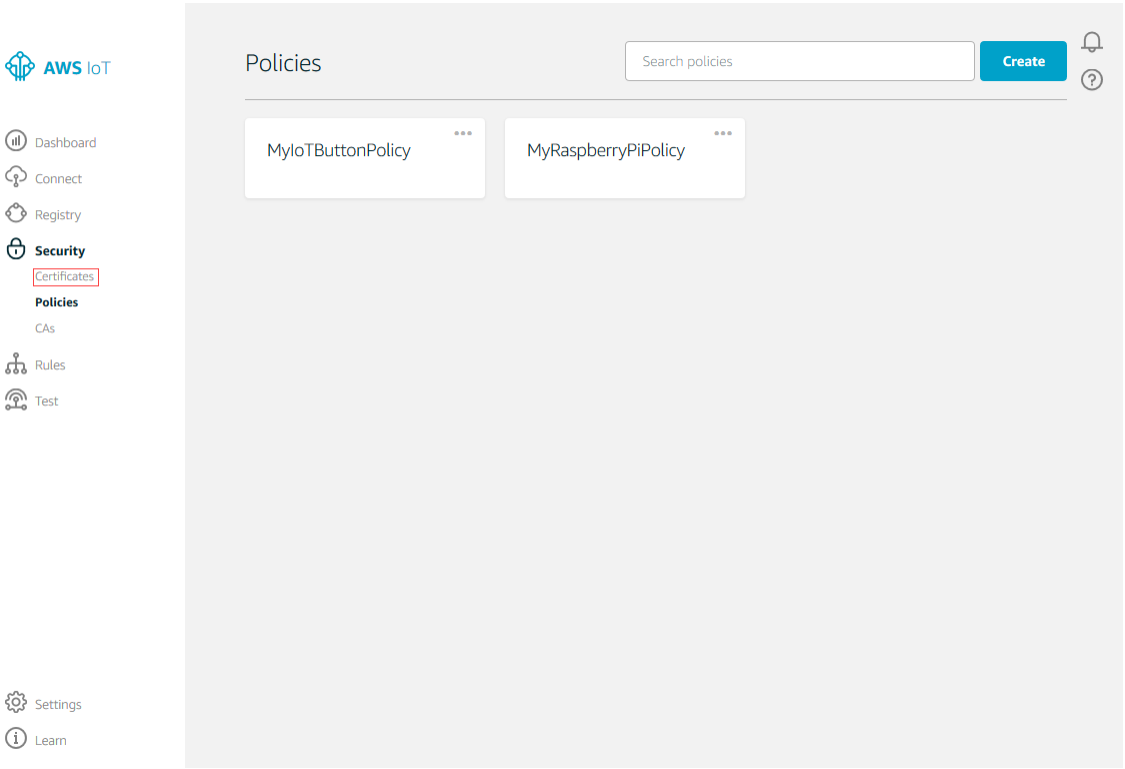
8. Choose **Create**.



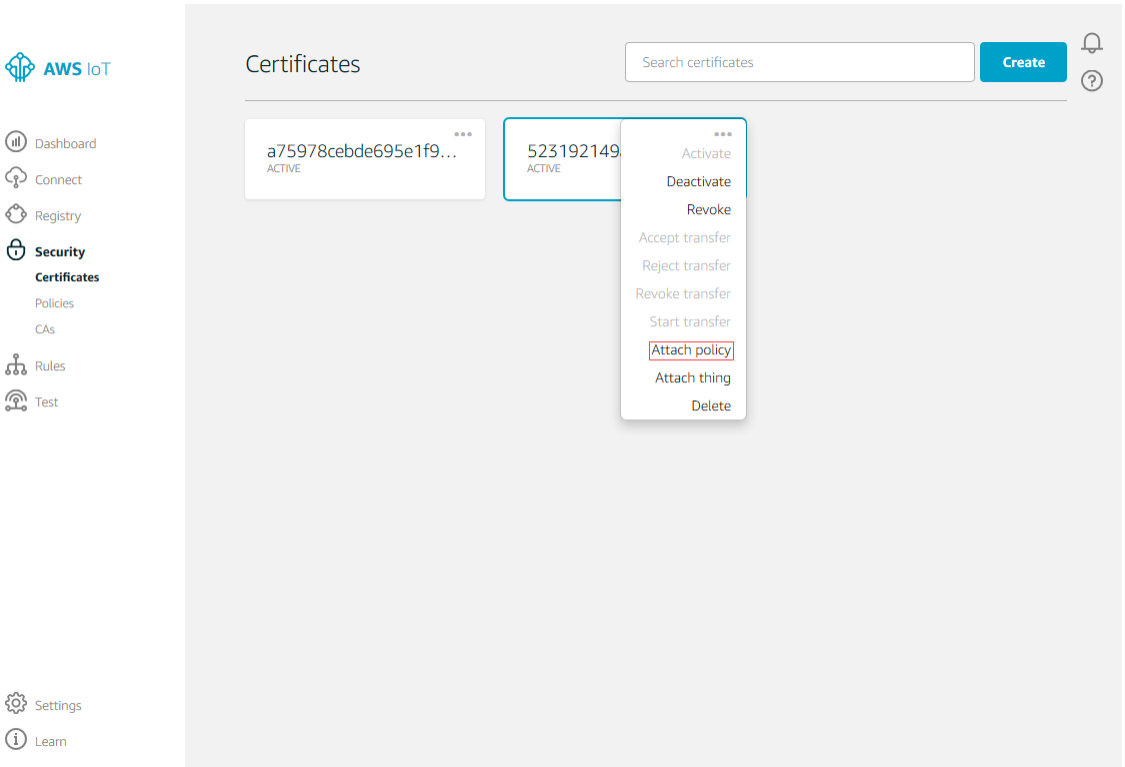
9. Choose the left arrow to return to the **Policies** page.



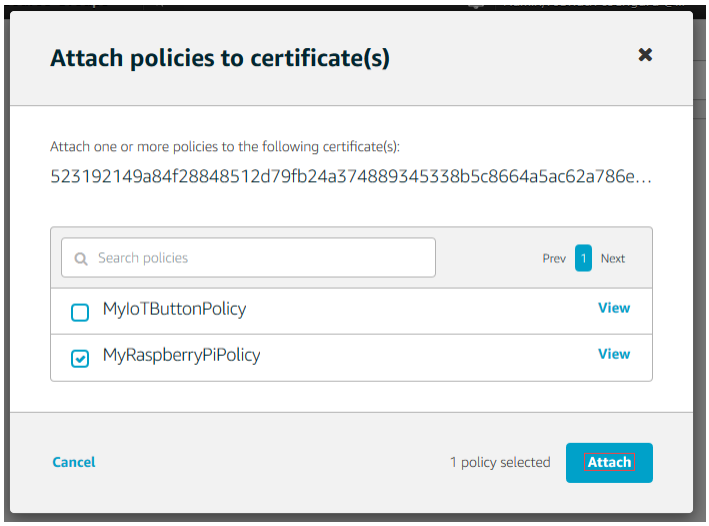
10. In the left navigation pane, under **Security**, choose **Certificates**.



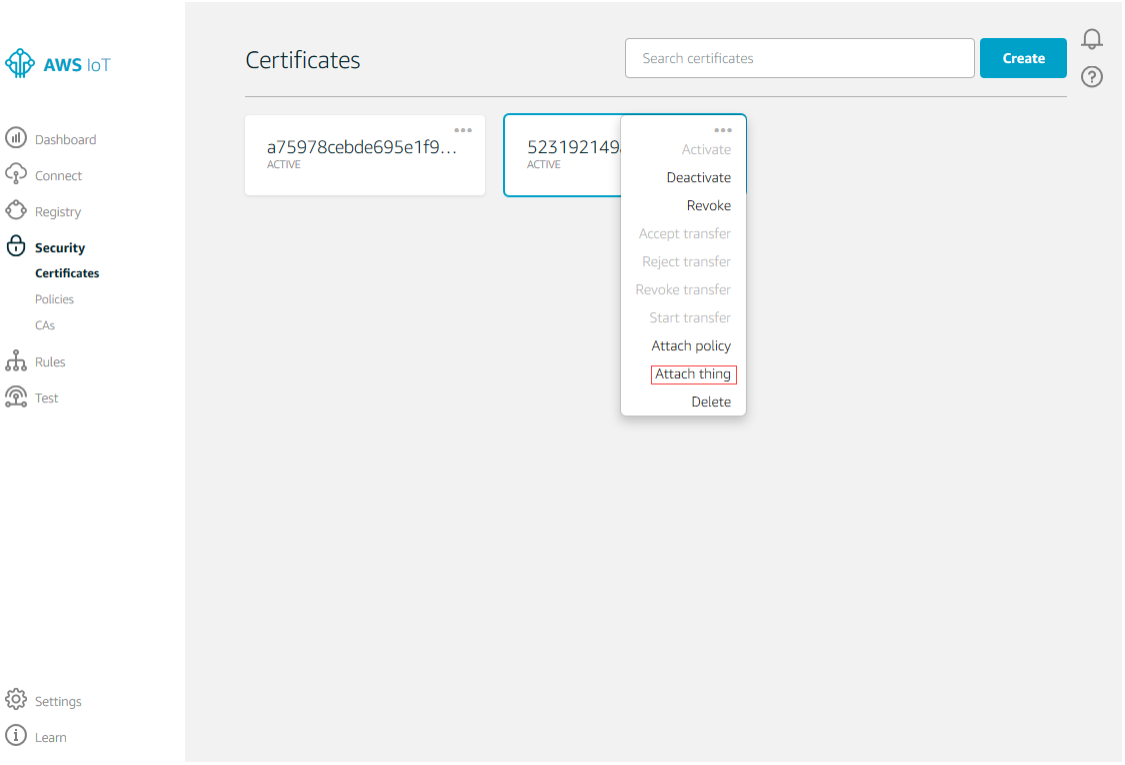
11. In the box for the certificate you created, choose **...** to open a drop-down menu, and then choose **Attach policy**.



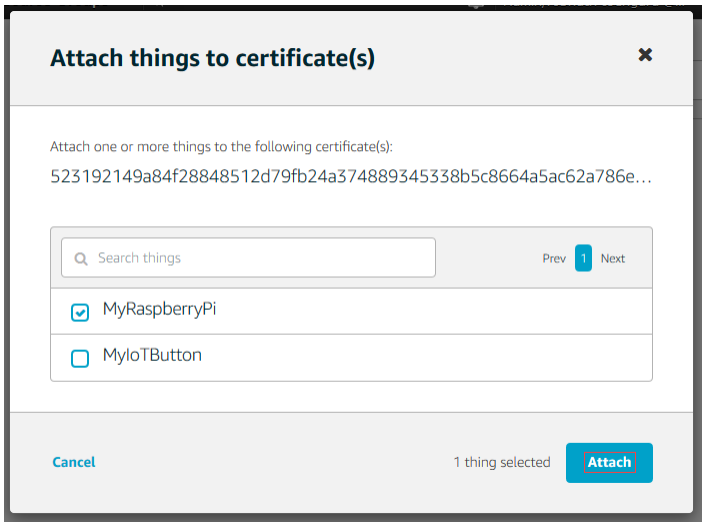
12. In the **Attach policies to certificate(s)** dialog box, select the check box next to the policy you created, and then choose **Attach**.



13. In the box for the certificate you created, choose **...** to open a drop-down menu, and then choose **Attach thing**.



14. In the **Attach things to certificate(s)** dialog box, select the check box next to the thing you created to represent your Raspberry Pi, and then choose **Attach**.



Using the AWS IoT Embedded C SDK

Set Up the Runtime Environment for the AWS IoT Embedded C SDK

1. Download the AWS IoT Device SDK for C from the following GitHub repository:

   ```
   git clone https://github.com/aws/aws-iot-device-sdk-embedded-C.git -b release
   ```

2. Before you can use the AWS IoT Embedded C SDK, you must download all required third-party libraries from GitHub. You can find instructions for doing this in the `deviceSDK/external_libs` folder.

Sample App Configuration

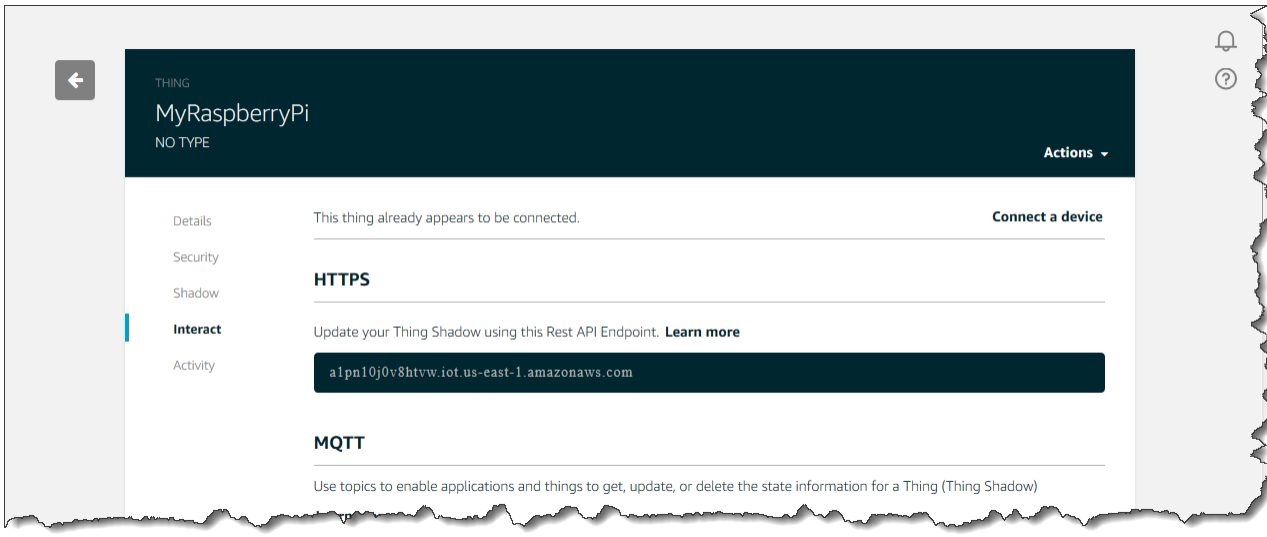
The AWS IoT Embedded C SDK includes sample apps for you to try. For simplicity, we are going to run subscribe_publish_sample.

1. Copy your certificate, private key, and root CA certificate into the `deviceSDK/certs` directory.

   If you did not get a copy of the root CA certificate, you can download it here. Copy the root CA text from the browser, paste it into a file, and then copy it into the `deviceSDK/certs` directory.

   Note
   Device and root CA certificates are subject to expiration or revocation. If this should occur, you must copy a new CA certificate or private key and device certificate onto your device.

2. Navigate to the `deviceSDK/sample_apps/subscribe_publish_sample` directory. You must configure your personal endpoint, private key, and certificate. The personal endpoint is the REST API endpoint you noted earlier. If you don't remember the endpoint and you have access to a machine with the AWS CLI installed, you can use the `aws iot describe-endpoint` command to find your personal endpoint URL. Or, go to the AWS IoT console. Choose **Registry**, choose **Things**, and then choose the thing that represents your Raspberry Pi. On the **Details** page for the thing, in the left navigation pane, choose **Interact**. Copy everything, including ".com", from **REST API endpoint**.



3. Open the `aws_iot_config.h` file and, in the `//Get from console` section, update the values for the following:

   AWS_IOT_MQTT_HOST

   Your personal endpoint.

   AWS_IOT_MY_THING_NAME

   Your thing name.

   AWS_IOT_ROOT_CA_FILENAME

   Your root CA certificate.

   AWS_IOT_CERTIFICATE_FILENAME

   Your certificate.

   AWS_IOT_PRIVATE_KEY_FILENAME

   Your private key.

   For example:

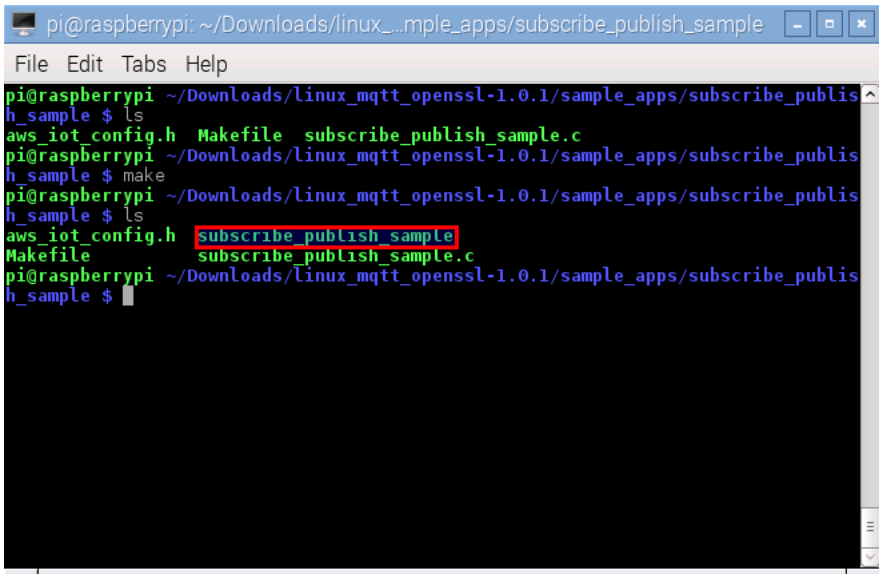
```
// Get from console
// ================================================
#define AWS_IOT_MQTT_HOST              "a22j5sm6o3yzc5.iot.us-east-1.amazonaws.com"
#define AWS_IOT_MQTT_PORT              8883
#define AWS_IOT_MQTT_CLIENT_ID         "MyRaspberryPi"
#define AWS_IOT_MY_THING_NAME          "MyRaspberryPi"
#define AWS_IOT_ROOT_CA_FILENAME       "root-CA.crt"
#define AWS_IOT_CERTIFICATE_FILENAME   "4bbdc778b9-certificate.pem.crt"
#define AWS_IOT_PRIVATE_KEY_FILENAME   "4bbdc778b9-private.pem.key"
// ================================================
```

Run Sample Applications

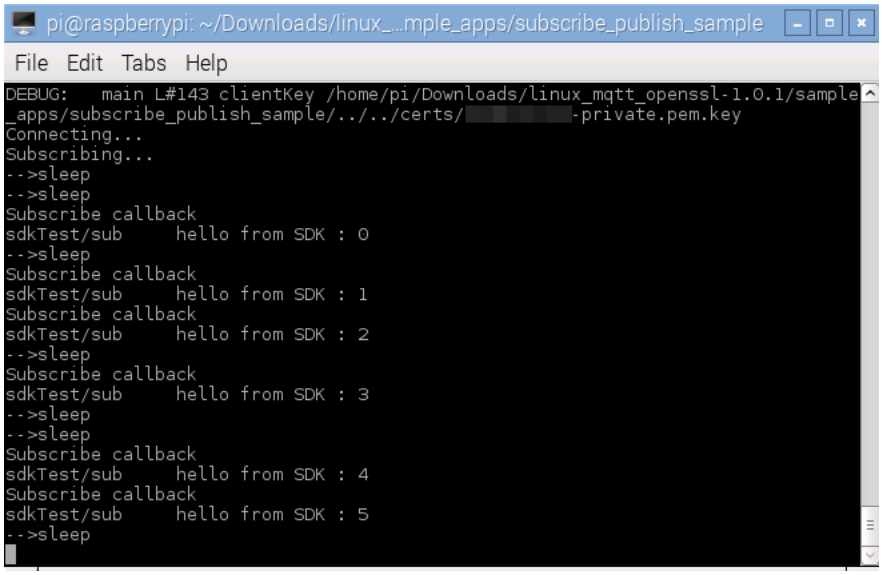
1. Compile the `subscribe_publish_sample_app` using the included makefile.

```
make -f Makefile
```

This generates an executable file.



2. Now run the subscribe_publish_sample_app. You should see output similar to the following:



Your Raspberry Pi is now connected to AWS IoT using the AWS IoT Device SDK for C.

Using the AWS IoT Device SDK for JavaScript

The easiest way to install the AWS IoT Device SDK for Node.js is to use `npm`. In this section we describe how to install Node and npm.

Set Up the Runtime Environment for the AWS IoT Device SDK for JavaScript

To use the AWS IoT Device SDK for JavaScript, install Node and the npm development tool on your Raspberry Pi. These packages are not installed by default.

Note
Before you continue, you might want to configure the keyboard mapping for your Raspberry Pi. For more information, see Configure Raspberry Pi Keyboard Mapping.

1. To add the Node repository, open a terminal and run the following command:

```
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
```

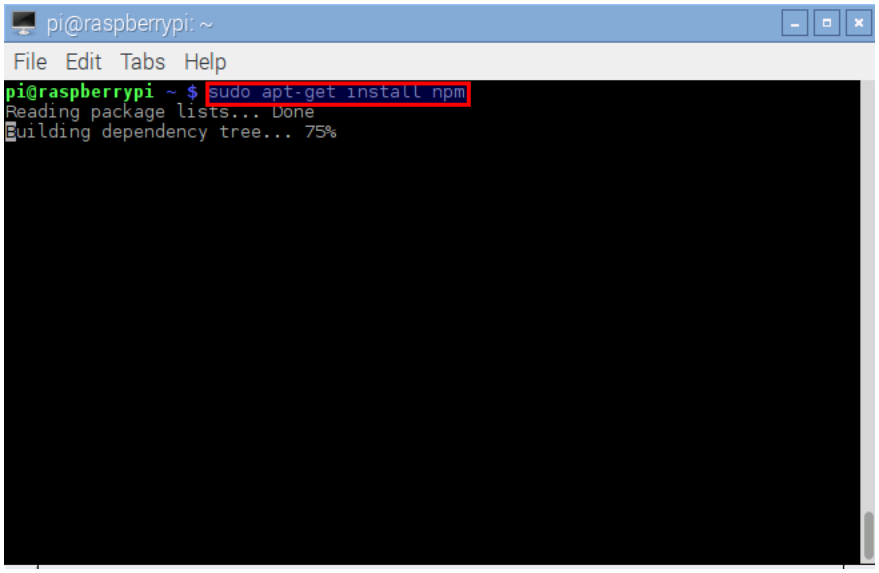
2. To install Node, run

```
sudo apt-get install nodejs
```

3. Run `npm -v` to determine if npm is install. If no version information is returned, install npm as follows:

```
sudo apt-get install npm
```

You should see output similar to the following:



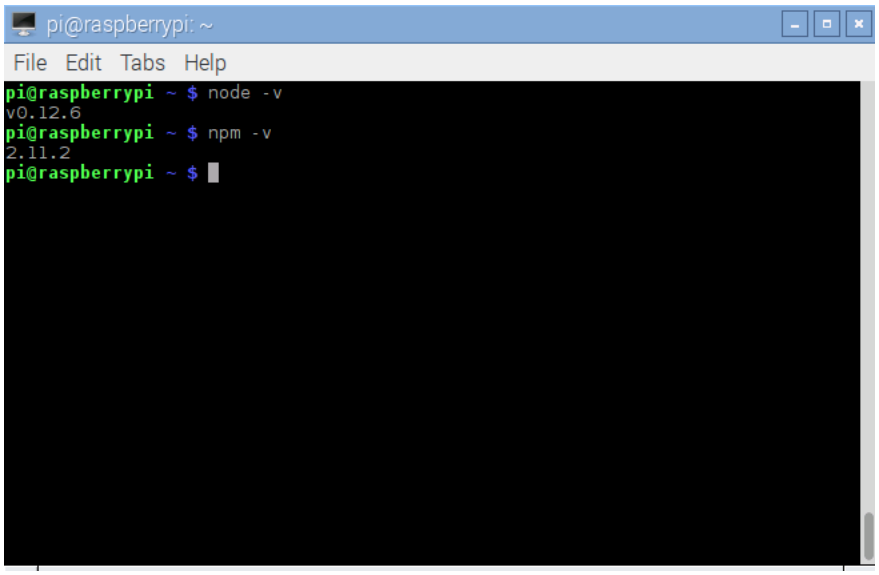
4. To verify the installation of Node and npm, run the following:

```
node -v
```

and

```
npm -v
```

You should see output similar to the following:



Install the AWS IoT Device SDK for JavaScript

To install the AWS IoT Device SDK for JavaScript/Node.js on your Raspberry Pi, open a console window and from your `~/deviceSDK` directory, use npm to install the SDK:

```
npm install aws-iot-device-sdk
```

After the installation is complete, you should find a `node_modules` directory in your `~/deviceSDK` directory.

Prepare to Run the Sample Applications

The AWS IoT Device SDK for JavaScript includes sample apps for you to try. To run them, you must configure your certificates and private key.

Edit the file ~/deviceSDK/node_modules/aws-iot-device-sdk/examples/lib/cmdline.js to change the default names for the private key (`privateKey`), certificate (`clientCert`), and CA root certificate (`caCert`) used by the samples. For example:

```
default: {
    region: 'us-east-1',
    clientId: clientIdDefault,
    privateKey: '4bbdc778b9-private.pem.key',
    clientCert: '4bbdc778b9-certificate.pem.crt',
    caCert: 'root-CA.crt,
    testMode: 1,
    reconnectPeriod: 3 * 1000,  /* milliseconds */
    delay: 4 * 1000      /* milliseconds */
};
```

Run the Sample Applications

Run the examples using

```
node examples/<YourDesiredExample>.js -f <certs location>
```

Assuming you are in the directory `~/deviceSDK/node_modules/aws-iot-device-sdk/`, the certificates location should be `~/deviceSDK/certs/`. For more information about how you can use command line options to specify the certificates location and your own host address, see Certificates.

If you want to create a configuration file for use with the command line option `--configuration-file (-F)`, create a file (in JSON format) with the following properties. For example:

```
{
    "host":           "a22j5sm6o3yzc5.iot.us-east-1.amazonaws.com",
    "port":           8883,
    "clientId":       "MyRaspberryPi",
    "thingName":      "MyRaspberryPi",
    "caCert":         "root-CA.crt",
    "clientCert":     "4bbdc778b9-certificate.pem.crt",
    "privateKey":     "4bbdc778b9-private.pem.key"
}
```

Your Raspberry Pi is now connected to AWS IoT using the AWS IoT SDK for JavaScript.

Managing Things with AWS IoT

AWS IoT provides a thing registry that helps you manage your things. A thing is a representation of a specific device or logical entity. It can be a physical device or sensor (for example, a light bulb or a switch on a wall). It can also be a logical entity like an instance of an application or physical entity that does not connect to AWS IoT but is related to other devices that do (for example, a car that has engine sensors or a control panel).

Information about a thing is stored in the thing registry as JSON data. Here is an example thing:

```
{
     "version": 3,
    "thingName": "MyLightBulb",
    "defaultClientId": "MyLightBulb",
    "thingTypeName": "LightBulb",
    "attributes": {
        "model": "123",
        "wattage": "75"
    }
}
```

Things are identified by a name. Things can also have attributes, which are name-value pairs you can use to store information about the thing, such as its serial number or manufacturer.

A typical device use case involves the use of the thing name as the default MQTT client ID. Although we do not enforce a mapping between a thing's registry name and its use of MQTT client IDs, certificates, or shadow state, we recommend you choose a thing name and use it as the MQTT client ID for both the thing registry and the Thing Shadows service. This provides organization and convenience to your IoT fleet without removing the flexibility of the underlying device certificate model or thing shadows.

You do not need to create a thing in the thing registry to connect it to AWS IoT. Adding your things in the thing registry allows you to manage and search for them more easily.

Managing Things with the Thing Registry

You use the AWS IoT console or the AWS CLI to interact with the registry. The following sections show how to use the CLI to work with the thing registry.

Create a thing

The following command shows how to use the AWS IoT `CreateThing` command from the CLI to create a thing:

```
$ aws iot create-thing --thing-name "MyLightBulb" --attribute-payload "{\"attributes\":
 {\"wattage\":\"75\", \"model\":\"123\"}}"
```

The `CreateThing` command will display the name and ARN of your new thing:

```
{
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
    "thingName": "MyLightBulb"
```

```
    "thingId": "12345678abcdefgh12345678ijklmnop12345678"
}
```

List things

You can use the `ListThings` command to list all things in your account:

```
$ aws iot list-things
{
    "things": [
       {
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MyLightBulb"
        },
        {
            "attributes": {
                "numOfStates":"3"
             },
            "version": 11,
            "thingName": "MyWallSwitch"
        }
    ]
}
```

Search for things

You can use the `DescribeThing` command to list information about a thing:

```
$ aws iot describe-thing --thing-name "MyLightBulb"
{
    "version": 3,
    "thingName": "MyLightBulb",
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
    "thingId": "12345678abcdefgh12345678ijklmnop12345678"
    "defaultClientId": "MyLightBulb",
    "thingTypeName": "StopLight",
    "attributes": {
        "model": "123",
        "wattage": "75"
    }
}
```

You can use the `ListThings` command to search for all things associated with a thing type name:

```
$  aws iot list-things --thing-type-name "LightBulb"
```

```
{
    "things": [
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
```

```
            },
            "version": 1,
            "thingName": "MyRGBLight"
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MySecondLightBulb"
        }
    ]
}
```

You can use the `ListThings` command to search for all things that have an attribute with a specific value:

```
$  aws iot list-things --attribute-name "wattage" --attribute-value "75"
```

```
{
    "things": [
        {
            "thingTypeName": "StopLight",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 3,
            "thingName": "MyLightBulb"
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MyRGBLight"
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MySecondLightBulb"
        }
    ]
}
```

Update a thing

You can use the `UpdateThing` command to update a thing:

```
$ aws iot update-thing --thing-name "MyLightBulb" --attribute-payload "{\"attributes\":
 {\"wattage\":\"150\", \"model\":\"456\"}}"
```

The `UpdateThing` command does not produce output. You can use the `DescribeThing` command to see the result:

```
$ aws iot describe-thing --thing-name "MyLightBulb"
{
    "attributes": {
        "model": "456",
        "wattage": "150"
    },
    "version": 2,
    "thingName": "MyLightBulb"
}
```

Delete a thing

You can use the `DeleteThing` command to delete a thing:

```
$ aws iot delete-thing --thing-name "MyThing"
```

Attach a principal to a thing

A physical device must have an X.509 certificate in order to communicate with AWS IoT. You can associate the certificate on your device with the thing in the thing registry that represents your device. To attach a certificate to your thing, use the `AttachThingPrincipal` command:

```
$ aws iot attach-thing-principal --thing-name "MyLightBulb" --principal "arn:aws:iot:us-
east-1:123456789012:cert/a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

The `AttachThingPrincipal` command does not produce any output.

Detach a principal from a thing

You can use the `DetachThingPrincipal` command to detach a certificate from a thing:

```
$ aws iot detach-thing-principal --thing-name "MyLightBulb" --principal "arn:aws:iot:us-
east-1:123456789012:cert/a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

The `DetachThingPrincipal` command does not produce any output.

Thing Types

Thing types allow you to store description and configuration information that is common to all things associated with the same thing type. This simplifies the management of things in the thing registry. For example, you can define a LightBulb thing type. All things associated with the LightBulb thing type share a set of attributes: serial number, manufacturer, and wattage. When you create a thing of type LightBulb (or change the type of an existing thing to LightBulb) you can specify values for each of the attributes defined in the LightBulb thing type.

Although thing types are optional, their use provides better discovery of things.

- Things with a thing type can have up to 50 attributes.
- Things without a thing type can have up to three attributes.
- A thing can only be associated with one thing type.

- There is no limit on the number of thing types you can create in your account.

Thing types are immutable. You cannot change a thing type name after it has been created. You can deprecate a thing type at any time to prevent new things from being associated with it. You can also delete thing types that have no things associated with them.

Create a Thing Type

You can use the `CreateThingType` command to create a thing type:

```
$ aws iot create-thing-type

                --thing-type-name "LightBulb" --thing-type-properties
 "thingTypeDescription=light bulb type, searchableAttributes=wattage,model"
```

The `CreateThingType` command returns a response that contains the thing type and its ARN:

```
{
    "thingTypeName": "LightBulb",
    "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb"
}
```

List thing types

You can use the `ListThingTypes` command to list thing types:

```
$ aws iot list-thing-types
```

The `ListThingTypes` command returns a list of the thing types defined in your AWS account:

```
{
    "thingTypes": [
        {
            "thingTypeName": "LightBulb",
            "thingTypeProperties": {
                "searchableAttributes": [
                    "wattage",
                    "model"
                ],
                "thingTypeDescription": "light bulb type"
            },
            "thingTypeMetadata": {
                "deprecated": false,
                "creationDate": 1468423800950
            }
        }
    ]
}
```

Describe a thing type

You can use the `DescribeThingType` command to get information about a thing type:

```
$ aws iot describe-thing-type --thing-type-name "LightBulb"
```

The `DescribeThingType` command responds with information about the specified type:

```
{
    "thingTypeName": "LightBulb",
    "thingTypeProperties": {
        "searchableAttributes": [
            "wattage",
            "model"
        ],
        "thingTypeDescription": "light bulb type"
    },
    "thingTypeMetadata": {
        "deprecated": false,
        "creationDate": 1468423800950,
    }
}
```

Associate a thing type with a thing

You can use the `CreateThing` command to specify a thing type when you create a thing:

```
$ aws iot create-thing --thing-name "MySecondLightBulb" --thing-type-name "LightBulb" --
attribute-payload "{\"attributes\": {\"wattage\":\"75\", \"model\":\"123\"}}"
```

You can use the `UpdateThing` command at any time to change the thing type associated with a thing:

```
$ aws iot update-thing --thing-name "MyLightBulb" --thing-type-name "StopLight" --
attribute-payload  "{\"attributes\": {\"wattage\":\"75\", \"model\":\"123\"}}"
```

You can also use the `UpdateThing` command to disassociate a thing from a thing type.

Deprecate a thing type

Thing types are immutable. They cannot be changed after they are defined. You can, however, deprecate a thing type to prevent users from associating any new things with it. All existing things associated with the thing type will be unchanged.

To deprecate a thing type, use the `DeprecateThingType` command:

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType"
```

You can use the `DescribeThingType` command to see the result:

```
$ aws iot describe-thing-type --thing-type-name "StopLight":
```

```
{
    "thingTypeName": "StopLight",
    "thingTypeProperties": {
        "searchableAttributes": [
            "wattage",
            "numOfLights",
            "model"
        ],
        "thingTypeDescription": "traffic light type",
    },
    "thingTypeMetadata": {
        "deprecated": true,
```

```
        "creationDate": 1468425854308,
        "deprecationDate": 1468446026349
    }
}
```

Deprecating a thing type is a reversible operation. You can undo a deprecation by using the `--undo-deprecate` flag with the `DeprecateThingType` CLI command:

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType" --undo-deprecate
```

You can use the `DescribeThingType` CLI command to see the result:

```
$ aws iot describe-thing-type --thing-type-name "StopLight":
```

```
{
    "thingTypeName": "StopLight",
    "thingTypeArn": "arn:aws:iot:us-east-1:123456789012:thingtype/StopLight",
    "thingTypeId": "12345678abcdefgh12345678ijklmnop12345678"
    "thingTypeProperties": {
        "searchableAttributes": [
            "wattage",
            "numOfLights",
            "model"
        ],
        "thingTypeDescription": "traffic light type"
    },
    "thingTypeMetadata": {
        "deprecated": false,
        "creationDate": 1468425854308,
    }
}
```

Delete a thing type

You can delete thing types only after they have been deprecated. To delete a thing type, use the `DeleteThingType` command:

```
$ aws iot delete-thing-type --thing-type-name "StopLight"
```

Note
You must wait five minutes after you deprecate a thing type before you can delete it.

Thing Groups

Thing groups allow you to manage several things at once by categorizing them into groups. Groups can also contain groups — you can build a hierarchy of groups. You can attach a policy to a parent group and it will be inherited by its child groups, and by all of the things in the group and in its child groups as well. This makes control of permissions easy for large numbers of things.

Here are the things you can do with thing groups:

- Create, describe or delete a group.
- Add a thing to a group, or to more than one group.
- Remove a thing from a group.
- List the groups you have created.

- List all child groups of a group (its direct and indirect descendants.)
- List the things in a group, including all the things in its child groups.
- List all ancestor groups of a group (its direct and indirect parents.)
- Add, delete or update the attributes of a group. (Attributes are name-value pairs you can use to store information about a group.)
- Attach or detach a policy to or from a group.
- List the policies attached to a group.
- List the policies inherited by a thing (by virtue of the policies attached to its group, or one of its parent groups.)
- Configure logging options for things in a group (see Configure AWS IoT Logging (p. 344).)
- Create jobs that will be sent to and executed on every thing in a group and its child groups (see Jobs (p. 263).)

Here are some limitations of thing groups:

- If a group will be a child of another group, this must be specified at the time it is created.
- You can't change a group's parent later. (So be sure to plan your group hierarchy and create a parent group before creating any child groups it will contain.
- You cannot add a thing to more than than 10 groups.
- You cannot add a thing to more than one group in the same hierarchy. (In other words, you cannot add a thing to two groups which share a common parent.)
- You cannot rename a group.

Attaching and detaching policies to groups can enhance the security of your AWS IoT operations in a number of significant ways. The per device method of attaching a policy to a certificate, which is then attached to a thing, is time consuming and makes it difficult to quickly update or change policies across a fleet of devices. Having a policy attached to the thing's group saves steps when it is time to rotate the certificates on a thing. And policies are dynamically applied to things when they change group membership, so you aren't required to re-create a complex set of permissions each time a device changes membership in a group.

Create a Thing Group

You can use the `CreateThingGroup` command to create a thing group:

```
$ aws iot create-thing-group --thing-group-name LightBulbs
```

The `CreateThingGroup` command returns a response that contains the thing group, its ID, and its ARN:

```
{
    "thingGroupName": "LightBulbs",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
}
```

Here is an example which specifies a parent of the thing group when it is created:

```
$ aws iot create-thing-group --thing-group-name RedLights --parent-group-name LightBulbs
```

As before, the `CreateThingGroup` command returns a response that contains the thing group and its ID and ARN:

```
{
    "thingGroupName": "RedLights",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
}
```

Important
Keep in mind the following limits when creating group hierarchies:

- A group can have at most one direct parent.
- A group may have no more than 100 direct child groups.
- The maximum depth of a group hierarchy is 7.
- A group can have up to 50 attributes. (Attributes are name-value pairs you can use to store information about a group.) Each attribute name can contain up to 128 characters and each value up to 800 characters.

Describe a thing group

You can use the `DescribeThingGroup` command to get information about a thing group:

```
$ aws iot describe-thing-group --thing-group-name RedLights
```

The `DescribeThingGroup` command responds with information about the specified group:

```
{
    "thingGroupName": "RedLights",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
    "thingGroupId": "12345678abcdefgh12345678ijklmnop12345678",
    "version": 1,
    "thingGroupMetadata": {
        "creationDate": 1478299948.882
        "parentGroupName": "Lights",
        "rootToParentThingGroups": [
            {
                "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/ShinyObjects",
                "groupName": "ShinyObjects"
            },
            {
                "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs",
                "groupName": "LightBulbs"
            }
        ]
    },
    "thingGroupProperties": {
        "attributePayload": {
            "attributes": {
                "brightness": "3400 lumens"
            },
        },
        "thingGroupDescription": "string"
    },
}
```

Add thing to thing group

You can use the `AddThingToThingGroup` command to add a thing to a group:

```
$ aws iot add-thing-to-thing-group --thing-name MyLightBulb --thing-group-name RedLights
```

The `AddThingToThingGroup` command does not produce any output.

Important
You can add a thing to a maximum of 10 groups. But you cannot add a thing to more than one group in the same hierarchy. (In other words, you cannot add a thing to two groups which share a common parent.)

Remove thing from thing group

You can use the `RemoveThingFromThingGroup` command to remove a thing from a group:

```
$ aws iot remove-thing-from-thing-group --thing-name MyLightBulb --thing-group-name
 RedLights
```

The `RemoveThingFromThingGroup` command does not produce any output.

List things in thing group

You can use the `ListThingsInThingGroup` command to list the things belonging to a group:

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs
```

The `ListThingsInThingGroup` command returns a list of the things in the given group:

```
{
    "things":[
        "TestThingA"
    ]
}
```

With the `--recursive` parameter, you can list things belonging to a group and those in any of its child groups as well:

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs --recursive
```

```
{
    "things":[
        "TestThingA",
        "MyLightBulb"
    ]
}
```

Note
This operation is eventually consistent, in other words, changes to the thing group may not be reflected immediately.

List thing groups

You can use the `ListThingGroups` command to list groups you have created:

```
$ aws iot list-thing-groups
```

The `ListThingGroups` command returns a list of the groups defined in your AWS account:

```
{
    "thingGroups": [
        {
            "groupName": "LightBulbs",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
        },
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "RedLEDLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLEDLights"
        },
        {
            "groupName": "RedIncandescentLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
RedIncandescentLights"
        }
        {
            "groupName": "ReplaceableObjects",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/ReplaceableObjects"
        }
    ]
}
```

Use the optional filters to list those groups that have a given group as parent (`--parent-group`) or groups whose name begins with a given prefix (`--name-prefix-filter`.) The `--recursive` parameter allows you to list all children groups as well, not just direct child groups of a thing group:

```
$ aws iot list-thing-groups --parent-group LightBulbs
```

In this case, the `ListThingGroups` command returns a list of the direct child groups of the thing group defined in your AWS account:

```
{
    "childGroups":[
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        }
    ]
}
```

Use the `--recursive` parameter with the `ListThingGroups` command to list *all* child groups of a thing group, not just direct children:

```
$ aws iot list-thing-groups --parent-group LightBulbs --recursive
```

The `ListThingGroups` command returns a list of all child groups of the thing group:

```
{
    "childGroups":[
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
```

```
        {
            "groupName": "RedLEDLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLEDLights"
        },
        {
            "groupName": "RedIncandescentLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
RedIncandescentLights"
        }
    ]
}
```

Note
This operation is eventually consistent, in other words, changes to the thing group may not be reflected immediately.

List groups for thing

You can use the `ListThingGroupsForThing` command to list the groups a thing belongs to, including any parent groups:

```
$ aws iot list-thing-groups-for-thing --thing-name MyLightBulb
```

The `ListThingGroupsForThing` command returns a list of the thing groups this thing belongs to, including any parent groups:

```
{
    "thingGroups":[
        {
            "groupName": "LightBulbs",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
        },
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "ReplaceableObjects",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/ReplaceableObjects"
        }
    ]
}
```

Update a Thing Group

You can use the `UpdateThingGroup` command to update the attributes of a thing group:

```
$ aws iot update-thing-group --thing-group-name "LightBulbs" --thing-group-properties
 "thingGroupDescription=\"this is a test group\", attributePayload=\"{\"attributes
\"={\"Owner\"=\"150\",\"modelNames\"=\"456\"}}"
```

The `UpdateThingGroup` command returns a response that contains the group's version number after the update:

```
{
    "version": 4
}
```

Note
A group can have up to 50 attributes.

Delete a thing group

To delete a thing group, use the `DeleteThingGroup` command:

```
$ aws iot delete-thing-group --thing-group-name "RedLights"
```

The `DeleteThingGroup` command does not produce any output.

Important
If you try to delete a thing group which has child thing groups, this will generate an error:

```
A client error (InvalidRequestException) occurred when calling the
 DeleteThingGroup
operation: Cannot delete thing group : RedLights when there are still child groups
 attached to it.
```

You must delete any child groups first before deleting the group.

You can delete a group that has child things, but any permissions granted to the things by membership in the group will no longer apply. Before deleting a group that has a policy attached, check carefully that removing those permissions would not stop the things in the group from being able to function properly. Also, note that commands that show which groups a thing belongs to (for example, `ListGroupsForThing`) may continue to show the group while records in the cloud are being updated.

Attach a policy to a thing group

You can use the `AttachPolicy` command to attach a policy to a thing group and so, by extension, to all things in that group and things in any of its child groups:

```
$ aws iot attach-policy \
  --target "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs" \
  --policy-name "myLightBulbPolicy"
```

The `AttachPolicy` command does not produce any output

Important
The maximum number of policies that can be attached to a group is 2.

The `--target` can also be a certificate ARN or an Amazon Cognito Identity. See Security and Identity for AWS IoT (p. 109) for more information about policies, certificates and authentication.

Detach a policy from a thing group

You can use the `DetachPolicy` command to detach a policy from a group and so, by extension, to all things in that group and things in any of its child groups:

```
$ aws iot detach-policy --target "LightBulbs" --policy-name "myLightBulbPolicy"
```

The `DetachPolicy` command does not produce any output.

List the policies attached to a thing group

You can use the `ListAttachedPolicies` command to list the policies attached to a group:

```
$ aws iot list-attached-policies --target "RedLights"
```

The `--target` parameter can also be a certificate ARN or Amazon Cognito Identity.

Add the optional `--recursive` parameter to include all policies attached to the group's parent groups as well.

The `ListAttachedPolicies` command returns a list of policies:

```
{
    "policies": [
        "MyLightBulbPolicy"
        ...
    ]
}
```

List the groups for a policy

You can use the `ListTargetsForPolicy` command to list the targets, including any groups, that a policy is attached to:

```
$ aws iot list-targets-for-policy --policy-name "MyLightBulbPolicy"
```

Add the optional `--page-size` *`number`* parameter to specify the maximum number of results to be returned for each query, and the `--marker` *`string`* parameter on subsequent calls to retrieve the next set of results, if any.

The `ListTargetsForPolicy` command returns a list of targets and the token to use to retrieve more results:

```
{
    "nextMarker": "string",
    "targets": [ "string" ... ]
}
```

Get effective policies for a thing

You can use the `GetEffectivePolicies` command to list the policies in effect for a thing, including the policies attached to any groups the thing belongs to (whether the group is a direct parent or indirect ancestor):

```
$ aws iot get-effective-policies \
  --thing-name "MyLightBulb" \
  --principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

Use the `--principal` parameter to specify the ARN of the certificate attached to the thing. If you are using Amazon Cognito Identity authentication, use the `--cognito-identity-pool-id` parameter and, optionally, add the `--principal` parameter to specify a specific Cognito Identity. (If you specify only the `--cognito-identity-pool-id` then the policies associated with that identity pool's role for unauthenticated users are returned. If you use both, the policies associated with that identity pool's role for authenticated users are returned.

The `--thing-name` parameter is optional and may be used instead of the `--principal` parameter. When used, the policies attached to any group the thing belongs to, and the policies attached to any parent groups of these groups (up to the root group in the hierarchy) will be returned.

The `GetEffectivePolicies` command returns a list of policies:

```
{
    "effectivePolicies": [
        {
            "policyArn": "string",
            "policyDocument": "string",
            "policyName": "string"
        }
        ...
    ]
}
```

Test authorization for MQTT actions

You can use the `TestAuthorization` command to test whether an MQTT action is allowed for a thing:

```
$ aws iot test-authorization \
  --principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847" \
  --auth-infos "[ \"actionType": \"PUBLISH\", \"resources\": [ \"arn:aws:iot:us-
east-1:123456789012:topic/iotButton/G030JF053216F1BS\" ] ]"
```

Use the `--principal` parameter to specify the ARN of the certificate attached to the thing. If using Amazon Cognito Identity authentication, specify a Cognito Identity as the `--principal` or use the `--cognito-identity-pool-id` parameter, or both. (If you specify only the `--cognito-identity-pool-id` then the policies associated with that identity pool's role for unauthenticated users are considered. If you use both, the policies associated with that identity pool's role for authenticated users are considered.

Specify one or more MQTT actions you want to test by listing sets of resources and action types following the `--auth-infos` parameter. The `actionType` field should contain "PUBLISH", "SUBSCRIBE", "RECEIVE", or "CONNECT". The `resources` field should contain a list of resource ARNs. See AWS IoT Policies (p. 122) for more information.

You can test the effects of adding policies by specifying them with the `--policy-names-to-add` parameter. Or you can test the effects of removing policies by them with the `--policy-names-to-skip` parameter.

You can use the optional `--client-id` parameter to further refine your results.

The `TestAuthorization` command returns details on actions which were allowed or denied for each set of `--auth-infos` queries you specified:

```
{
    "authResults": [
        {
            "allowed": {
                "matchedStatements": [
                    {
                        "endPosition": {
                            "column": number,
                            "line": number
                        },
                        "policy": {
                            "policyArn": "string",
                            "policyName": "string"
                        },
                        "startPosition": {
```

```
                            "column": number,
                            "line": number
                        }
                    }
                ],
                "policies": [
                    {
                        "policyArn": "string",
                        "policyName": "string"
                    }
                ]
            },
            "authDecision": "string",
            "authInfo": {
                "actionType": "string",
                "resources": [ "string" ]
            },
            "denied": {
                "explicitDeny": {
                    "matchedStatements": [
                        {
                            "endPosition": {
                                "column": number,
                                "line": number
                            },
                            "policy": {
                                "policyArn": "string",
                                "policyName": "string"
                            },
                            "startPosition": {
                                "column": number,
                                "line": number
                            }
                        }
                    ],
                    "policies": [
                        {
                            "policyArn": "string",
                            "policyName": "string"
                        }
                    ]
                },
                "implicitDeny": {
                    "policies": [
                        {
                            "policyArn": "string",
                            "policyName": "string"
                        }
                    ]
                }
            },
            "missingContextValues": [ "string" ]
        }
    ]
}
```

Security and Identity for AWS IoT

Each connected device must have a credential to access the message broker or the Thing Shadows service. All traffic to and from AWS IoT must be encrypted over Transport Layer Security (TLS). Device credentials must be kept safe in order to send data securely to the message broker. AWS Cloud security mechanisms protect data as it moves between AWS IoT and other devices or AWS services.

- You are responsible for managing device credentials (X.509 certificates, AWS credentials) on your devices and policies in AWS IoT. You are responsible for assigning unique identities to each device and managing the permissions for a device or group of devices.
- Devices connect using your choice of identity (X.509 certificates, IAM users and groups, Amazon Cognito identities, or custom authentication tokens) over a secure connection according to the AWS IoT connection model.
- When using AWS IoT authentication, the message broker authenticates and authorizes all actions in your account. The message broker is responsible for authenticating your devices, securely ingesting device data, and adhering to the access permissions you place on devices using policies.
- When using custom authentication, a custom authorizer is responsible for authenticating your devices and providing an AWS IoT/IAM policy to authorize actions in your account.
- The AWS IoT rules engine forwards device data to other devices and other AWS services according to rules you define. It uses AWS access management systems to securely transfer data to its final destination.

AWS IoT Authentication

AWS IoT supports four types of identity principals for authentication:

- X.509 certificates
- IAM users, groups, and roles
- Amazon Cognito identities
- Federated identities

These identities can be used with mobile applications, web applications, or desktop applications. They can even be used by a user typing AWS IoT CLI commands. Typically, AWS IoT devices use X.509

certificates, while mobile applications use Amazon Cognito identities. Web and desktop applications use IAM or federated identities. CLI commands use IAM.

X.509 Certificates

X.509 certificates are digital certificates that use the X.509 public key infrastructure standard to associate a public key with an identity contained in a certificate. X.509 certificates are issued by a trusted entity called a certification authority (CA). The CA maintains one or more special certificates called CA certificates that it uses to issue X.509 certificates. Only the certification authority has access to CA certificates.

> **Note**
> We recommend that each device be given a unique certificate to enable fine-grained management including certificate revocation.
> Devices must support rotation and replacement of certificates in order to ensure smooth operation as certificates expire.

AWS IoT supports the following certificate-signing algorithms:

- SHA256WITHRSA
- SHA384WITHRSA
- SHA384WITHRSA
- SHA512WITHRSA
- RSASSAPSS
- DSA_WITH_SHA256
- ECDSA-WITH-SHA256
- ECDSA-WITH-SHA384
- ECDSA-WITH-SHA512

Certificates provide several benefits over other identification and authentication mechanisms. Certificates enable asymmetric keys to be used with devices. This means you can burn private keys into secure storage on a device. This way, sensitive cryptographic material never leaves the device. Certificates provide stronger client authentication over other schemes, such as user name and password or bearer tokens, because the secret key never leaves the device.

AWS IoT authenticates certificates using the TLS protocol's client authentication mode. TLS is available in many programming languages and operating systems and is commonly used for encrypting data. In TLS client authentication, AWS IoT requests a client X.509 certificate and validates the certificate's status and AWS account against a registry of certificates. It then challenges the client for proof of ownership of the private key that corresponds to the public key contained in the certificate.

To use AWS IoT certificates, clients must support all of the following in their TLS implementation:

- TLS 1.2.
- SHA-256 RSA certificate signature validation.
- One of the cipher suites from the TLS cipher suite support section.

X.509 Certificates and AWS IoT

AWS IoT can use AWS IoT-generated certificates or certificates signed by a CA certificate for device authentication. Certificates generated by AWS IoT do not expire. The expiry date and time for certificates signed by a CA certificate are set when the certificate is created.

> **Note**
> We recommend that each device be given a unique certificate to enable fine-grained management including certificate revocation.
> Devices must support rotation and replacement of certificates in order to ensure smooth operation as certificates expire.

To use a certificate that is not created by AWS IoT, you must register a CA certificate. All device certificates must be signed by the CA certificate you register.

You can use the AWS IoT console or CLI to perform the following operations:

- Create and register an AWS IoT certificate.
- Register a CA certificate.
- Register a device certificate.
- Activate or deactivate a device certificate.
- Revoke a device certificate.
- Transfer a device certificate to another AWS account.
- List all CA certificates registered to your AWS account.
- List all device certificates registered to your AWS account.

For more information about the CLI commands to use to perform these operations, see AWS IoT CLI Reference.

For more information about using the AWS IoT console to create certificates, see Create and Activate a Device Certificate.

Server Authentication

Server certificates allow your devices to verify that they're communicating with AWS IoT and not another server impersonating AWS IoT. AWS IoT server certificates are signed by one of the following CA certificates:

- RSA 2048 bit key: VeriSign Class 3 Public Primary G5 root CA certificate
- RSA 2048 bit key: Amazon Root CA 1
- RSA 4096 bit key: Amazon Root CA 2
- ECC 256 bit key: Amazon Root CA 3
- ECC 384 bit key: Amazon Root CA 4

In order for your devices to validate the AWS IoT server certificate we recommend installing all of the CA certificates listed above on your devices.

Storing all of these certificates on your device can take up valuable memory space. If your devices implement RSA-based validation, you can omit the Amazon Root CA 3 and Amazon Root CA 4 ECC certificates. If your devices implement ECC-based certificate validation, you can omit the Amazon Root CA 1 and Amazon Root CA 2 RSA certificates.

You will need to include the VeriSign Class 3 Public Primary G5 root CA certificate regardless of the type of certificate validation your devices use.

> **Note**
> CA certificates have an expiration date after which they cannot be used to validate a server's certificate. CA certificates may need to be replaced prior to their expiration date. You should

ensure that you can update the root CA certificates on all of your devices to ensure ongoing connectivity and to keep up to date with security best practices.

Reference the CA root certificate in your device code when you connect to AWS IoT. For more information, see the AWS IoT Device SDKs (p. 328).

Create and Register an AWS IoT Device Certificate

You can use the AWS IoT console or the AWS IoT CLI to create an AWS IoT certificate.

To create a certificate (console)

1. Sign in to the AWS Management Console and open the AWS IoT console at https://console.aws.amazon.com/iot.
2. In the left navigation pane, choose **Security** to expand the choices, and then choose **Certificates**. Choose **Create**.
3. Choose **One-click certificate creation** - **Create certificate**. Alternatively, to generate a certificate with a certificate signing request (CSR), choose **Create with CSR**.
4. Use the links to the public key, private key, and certificate to download each to a secure location.
5. Choose **Activate**.

To create a certificate (CLI)

The AWS IoT CLI provides two commands to create certificates:

- create-keys-and-certificate

  The CreateKeysAndCertificate API creates a private key, public key, and X.509 certificate.
- create-certificate-from-csr

  The CreateCertificateFromCSR API creates a certificate given a CSR.

Use Your Own Certificate

To use your own X.509 certificates, you must register a CA certificate with AWS IoT. The CA certificate can then be used to sign device certificates. You can register up to 10 CA certificates with the same subject field per AWS account per region. This allows you to have more than one CA sign your device certificates.

> **Note**
> Device certificates must be signed by the registered CA certificate. It is common for a CA certificate to be used to create an intermediate CA certificate. If you are using an intermediate certificate to sign your device certificates, you must register the intermediate CA certificate. Use the AWS IoT root CA certificate when you connect to AWS IoT even if you register your own root CA certificate. The AWS IoT root CA certificate is used by a device to verify the identity of the AWS IoT servers.

Contents

- Revoke the Device Certificate (p. 116)

If you do not have a CA certificate, you can use OpenSSL tools to create one.

To create a CA certificate

1. Generate a key pair.

```
openssl genrsa -out rootCA.key 2048
```

2. Use the private key from the key pair to generate a CA certificate.

```
openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem
```

Registering Your CA Certificate

To register your CA certificate, you must:

- Get a registration code from AWS IoT.
- Sign a private key verification certificate with your CA certificate.
- Pass your CA certificate and a private key verification certificate to the `register-ca-certificate` CLI command.

  The `Common Name` field in the private key verification certificate must be set to the registration code generated by the `get-registration-code` CLI command. A single registration code is generated per AWS account. You can use the `register-ca-certificate` command or the AWS IoT console to register CA certificates.

To register a CA certificate

1. Get a registration code from AWS IoT. This code will be used as the `Common Name` of the private key verification certificate.

```
aws iot get-registration-code
```

2. Generate a key pair for the private key verification certificate.

```
openssl genrsa -out verificationCert.key 2048
```

3. Create a CSR for the private key verification certificate. Set the `Common Name` field of the certificate to your registration code.

```
openssl req -new -key verificationCert.key -out verificationCert.csr
```

   You will be prompted for some information, including the `Common Name`, for the certificate.

```
Country Name (2 letter code) [AU]:
State or Province Name (full name) []:
Locality Name (for example, city) []:
Organization Name (for example, company) []:
Organizational Unit Name (for example, section) []:
Common Name (e.g. server FQDN or YOUR name) []:XXXXXXXXXXXXMYREGISTRATIONCODEXXXXXX
Email Address []:
```

4. Use the CSR to create a private key verification certificate.

```
openssl x509 -req -in verificationCert.csr -CA rootCA.pem -CAkey rootCA.key -
CAcreateserial -out verificationCert.pem -days 500 -sha256
```

5. Register the CA certificate with AWS IoT. Pass in the CA certificate and the private key verification certificate to the `register-ca-certificate` CLI command.

```
aws iot register-ca-certificate --ca-certificate file://rootCA.pem --verification-cert
 file://verificationCert.pem
```

6. Use the `update-certificate` CLI command to activate the CA certificate .

```
aws iot update-ca-certificate --certificate-id xxxxxxxxxxx --new-status ACTIVE
```

Creating a Device Certificate Using Your CA Certificate

You can use a CA certificate registered with AWS IoT to create a device certificate. The device certificate must be registered with AWS IoT before use.

To create a device certificate

1. Generate a key pair.

```
openssl genrsa -out deviceCert.key 2048
```

2. Create a CSR for the device certificate.

```
openssl req -new -key deviceCert.key -out deviceCert.csr
```

You will be prompted for some information, as shown here.

```
Country Name (2 letter code) [AU]:
State or Province Name (full name) []:
Locality Name (for example, city) []:
Organization Name (for example, company) []:
Organizational Unit Name (for example, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

3. Create a device certificate from the CSR.

```
openssl x509 -req -in deviceCert.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -
out deviceCert.pem -days 500 -sha256
```

> **Note**
> You must use the CA certificate registered with AWS IoT to create device certificates. If you have more than one CA certificate (with the same subject field and public key) registered in your AWS account, you must specify the CA certificate used to create the device certificate when you register your device certificate.

4. Register a device certificate.

```
aws iot register-certificate --certificate-pem file://deviceCert.pem --ca-certificate-
pem file://rootCA.pem
```

5. Use the `update-certificate` CLI command to activate the device certificate .

```
aws iot update-certificate --certificate-id xxxxxxxxxxx --new-status ACTIVE
```

Registering a Device Certificate

You must use the CA certificate registered with AWS IoT to sign device certificates. If you have more than one CA certificate (with the same subject field and public key) registered in your AWS account, you must specify the CA certificate used to sign the device certificate when you register your device certificate. You can register each device certificate manually, or you can use automatic registration, which allows devices to register their certificate when they connect to AWS IoT for the first time.

Registering Device Certificates Manually

Use the following CLI command to register a device certificate:

```
aws iot register-certificate --certificate-pem file://deviceCert.crt --ca-certificate-pem
 file://caCert.crt
```

Using Automatic/Just-in-Time Registration for Device Certificates

To register device certificates automatically when devices first connect to AWS IoT, you must enable automatic registration for your CA certificate. This will register any device certificate signed by your CA certificate when it connects to AWS IoT.

Enable Automatic Registration

Use the `update-ca-certificate` API to set the `auto-registration-status` of the CA certificate to `ENABLE`:

```
$ aws iot update-ca-certificate --cert-id caCertificateId --new-auto-registration-status
 ENABLE
```

You can also set the `auto-registration-status` to `ENABLE` when you use the `register-ca-certificate` API to register your CA certificate:

```
aws iot register-ca-certificate --ca-certificate file://rootCA.pem --verification-cert
 file://privateKeyVerificationCert.crt --allow-auto-registration
```

When a device first attempts to connect to AWS IoT, as part of the TLS handshake, it must present a registered CA certificate and a device certificate. AWS IoT recognizes the CA certificate as a registered CA certificate and automatically registers the device certificate and sets its status to `PENDING_ACTIVATION`. This means the device certificate was automatically registered and is awaiting activation. A certificate must be in the ACTIVE state before it can be used to connect to AWS IoT. When AWS IoT automatically registers a certificate or when a certificate in PENDING_ACTIVATION status connects, AWS IoT publishes a message to the following MQTT topic:

```
$aws/events/certificates/registered/caCertificateID
```

Where `caCertificateID` is the ID of the CA certificate that issued the device certificate.

The message published to this topic has the following structure:

```
{
    "certificateId": "certificateID",
    "caCertificateId": "caCertificateId",
    "timestamp": timestamp,
```

```
    "certificateStatus": "PENDING_ACTIVATION",
    "awsAccountId": "awsAccountId",
    "certificateRegistrationTimestamp": "certificateRegistrationTimestamp"
}
```

You can create a rule that listens on this topic and performs some actions. We recommend that you create a Lambda rule that verifies the device certificate is not on a certificate revocation list (CRL), activates the certificate, and creates and attaches a policy to the certificate. The policy determines which resources the device is able to access. For more information about how to create a Lambda rule that listens on the `$aws/events/certificates/registered/caCertificateID` topic and performs these actions, see Just-in-Time Registration.

Deactivate the CA Certificate

When you register a device certificate, AWS will check if the associated CA certificate is `ACTIVE`. If the CA certificate is `INACTIVE`, AWS IoT does not allow the device certificate to be registered. By marking the CA certificate as INACTIVE, you prevent any new device certificates issued by the compromised CA to be registered in your account. You can use the `update-ca-certificate` API to deactivate the CA certificate:

```
$ aws iot update-ca-certificate --cert-id certificateId --new-status INACTIVE
```

> **Note**
> Any registered device certificates that were signed by the compromised CA certificate will continue to work until you explicitly revoke them.

Use the `ListCertificatesByCA` API to get a list of all registered device certificates that were signed by the compromised CA. For each device certificate signed by the compromised CA certificate, use the `UpdateCertificate` API to revoke the device certificate to prevent it from being used.

Revoke the Device Certificate

If you detect suspicious activity on a registered device certificate, you can use the `update-certificate` API to revoke it:

```
$ aws iot update-certificate --cert-id certificateId
                             --new-status REVOKED
```

If any error or exception occurs during the auto-registration of the device certificates, AWS IoT sends events or messages to your logs in CloudWatch Logs. For more information about setting up the logs for your account, see the Amazon CloudWatch documentation.

IAM Users, Groups, and Roles

IAM users, groups, and roles are the standard mechanisms for managing identity and authentication in AWS. You can use them to connect to AWS IoT HTTP interfaces using the AWS SDK and CLI.

IAM roles also allow AWS IoT to access other AWS resources in your account on your behalf. For example, if you want to have a device publish its state to a DynamoDB table, IAM roles allow AWS IoT to interact with Amazon DynamoDB. For more information, see IAM Roles.

For message broker connections over HTTP, AWS IoT authenticates IAM users, groups, and roles using the Signature Version 4 signing process. For information, see Signing AWS API Requests.

When using AWS Signature Version 4 with AWS IoT, clients must support the following in their TLS implementation:

- TLS 1.2, TLS 1.1, TLS 1.0.

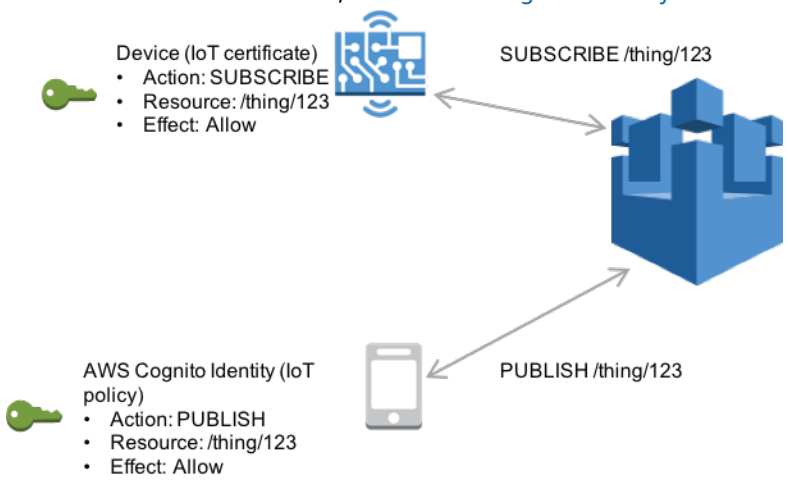
- SHA-256 RSA certificate signature validation.
- One of the cipher suites from the TLS cipher suite support section.

For information, see the IAM User Guide.

Amazon Cognito Identities

Amazon Cognito Identity allows you to use your own identity provider or other popular identity providers, such as Login with Amazon, Facebook, or Google. You exchange a token from your identity provider for AWS security credentials. The credentials represent an IAM role and can be used with AWS IoT.

AWS IoT extends Amazon Cognito and allows policy attachment to Amazon Cognito identities. You can attach a policy to an Amazon Cognito identity and give fine-grained permissions to an individual user of your AWS IoT application. In this way, you can assign permissions between specific customers and their devices. For more information, see Amazon Cognito Identity.



Custom Authentication

AWS IoT allows you to define custom authorizers that allow you to manage your own authentication and authorization strategy using a custom authentication service and a Lambda function. Custom authorizers allow AWS IoT to authenticate your devices and authorize operations using bearer token authentication and authorization strategies.

When an HTTP connection is established (and optionally upgraded to a WebSocket connection) and SigV4 headers are not present, the AWS IoT device gateway checks if a custom authorizer is configured for the endpoint, and if so, it is used to authenticate the connection and authorize the device. Custom authorizers can implement various authentication strategies (for example: JWT verification, OAuth provider call out, and so on) and must return policy documents which are used by the device gateway to authorize MQTT operations.

Custom Authorizers

Custom authorizers consist of:

A name

A unique arbitrary string that identifies the authorizer.

A Lambda function ARN

An ARN of a Lambda function that implements the authentication logic and returns authorization policies.

A public key

The public key from a key pair that is used to prevent unauthorized calls to the authorizer's Lambda function.

Use the following command to generate a key pair: `openssl genrsa -out myKeyPair.pem 2048`. Use the following command to extract the public key from the key pair: `openssl rsa -in myKeyPair.pem -pubout > mykey.pub`

Token key name

The key name used to extract tokens from the WebSocket connection headers.

The logic that performs the authentication is implemented in a Lambda function.

Note
AWS Lambda usage is billed. For more information about Lambda, see AWS Lambda Developer Guide.

This function takes a token presented by a device, authenticates the device, and returns the following information:

isAuthenticated

A Boolean value that indicates whether the token was authenticated. If this is `false`, the rest of the response fields should be ignored.

policy

A list of policies that specifies which operations the token bearer can perform.

DisconnectAfterInSecs

The length of time, in seconds, to keep the WebSocket connection open.

RefreshAfterInSecs

The length of time, in seconds, after which the Lambda function is invoked to refresh the policies.

Context

Additional information derived after validating the token. This information is made available in AWS IoT rules engine SQL statements and IAM/AWS IoT policy variables.

You must grant permission to the AWS IoT service principal to invoke the Lambda function that implements the custom authentication/authorization logic. You can do this with the following CLI command:

```
aws lambda add-permission --function-name <authorizer-function-name>
    --statement-id <unique_identifier_string>
    --action 'lambda:*'
    --principal iot.amazonaws.com
    --source-arn arn:aws:iot:<your-aws-region>:<account_id>:authorizer/<authorizer-
function-name>
    --region <your-aws-region>
```

For more information about granting permission to call Lambda functions, see AWS Lambda Permissions

You can set a default authorizer that is used when authorizer information is not included in a connection request:

```
aws iot set-default-authorizer --authorizer-name <my-authorizer>
    --region <your-aws-region>
    --endpoint <your-iot-endpoint
```

The `--source-arn` parameter ensures your Lambda function can only be invoked by the intended custom authorizer.

Configure a Custom Authorizer

1. Create a Lambda function that implements your authentication/authorization logic.
2. Register a custom authorizer with AWS IoT using the `Create-Authorizer` API.

```
aws iot create-authorizer --authorizer-name MyAuthorizer
    --authorizer-function-arn arn:aws:lambda:us-
west-2:<account_id>:function:MyAuthorizerFunction   // Lambda ARN
    --token-key-name MyAuthorizerToken
    // Key use to extract token from headers
    --token-signing-public-keys FIRST_KEY=
    // Public key used to verify token signature
"-----BEGIN PUBLIC KEY-----
[...insert your public key here...]
-----END PUBLIC KEY-----"
    --status ACTIVE
    // Authorizer status - must be ACTIVE
    --region us-west-2
    // AWS region
    --endpoint https://us-west-2.iot.amazonaws.com
    // IoT endpoint
```

Custom Authorizer Workflow

For a device to authenticate with the AWS IoT device gateway using a custom authorizer, it needs both a token and a signature used by AWS to validate the tokens before invoking the authorizer.

When a device attempts to connect to AWS IoT, it sends the following information in HTTP headers:

- A token generated by your authentication service.
- The signature generated by your authentication service.
- The authorizer used to authenticate the token. If omitted, the default authorizer is used.

The following is an example HTTP request to connect to AWS IoT over the WebSocket protocol.

```
GET /mqtt HTTP/1.1
Host: <your-iot-endpoint>
Upgrade: WebSocket
Connection: Upgrade
x-amz-customauthorizer-name = <authorizer-name>
x-amz-customauthorizer-signature = <token-signature>
<token-key-name> = <some-token>
```

In this example, the `x-amz-customauthorizer-name` header specifies the custom authorizer to use, the `x-amz-customauthorizer-signature` header contains the digital signature used to verify the

token, and the `token-key-name` is the token key name specified by the --token-key-name passed to the create-authorizer API.

The AWS IoT device gateway validates the digital signature and if valid, calls the specified authorizer. The following is an example payload AWS IoT sends to the custom authenticator's Lambda function.

```
{
   "type":"TOKEN",
   "authorizationToken":"<caller-supplied-token>",
   "authorizerId":<authorizer-id>,
   "endpoint":"<your-iot-endpoint>"
}
```

The authorizer validates the token and returns a principal ID, its associated AWS IoT/IAM policy, time-to-live (TTL) information for the connection, and any additional context generated by the authorizer.

The following is an example of the response from a custom authorizer.

```
{
 "isAuthenticated":true,
 "principalId": "xxxxxxxx",
 "policyDocument": {
   "Version": "2012-10-17",
   "Statement": [
     {
       "Action": "...",
       "Effect": "Allow|Deny",
       "Resource": "..."
     }
   ]
 },
 "connectionTtl:300,
 "context": {
   "username" : "foo",
   "city" : "Seattle",
   "country" : "USA"
  }
}
```

The AWS IoT device gateway then establishes the WebSocket connection. AWS IoT caches the policies associated with the principal so subsequent calls can be authorized without having to reauthenticate the device. Any failure that occurs during custom authentication results in authentication failure and connection termination.

Authorization

Policies determine what an authenticated identity can do. An authenticated identity is used by devices, mobile applications, web applications, and desktop applications. An authenticated identity can even be a user typing AWS IoT CLI commands. The identity can execute AWS IoT operations only if it has a policy that grants it permission.

Both AWS IoT policies and IAM policies are used with AWS IoT to control the operations an identity (also called a *principal*) can perform. The policy type you use depends on the type of identity you are using to authenticate with AWS IoT. The following table shows the identity types, the protocols they use, and the policy types that can be used for authorization.

AWS IoT operations are divided into two groups:

- Control plane API allows you to perform administrative tasks like creating or updating certificates, things, rules, and so on.
- Data plane API allows you send data to and receive data from AWS IoT.

The type of policy you use depends on whether you are using control plane or data plane API.

AWS IoT Data Plane API and Policy Types

Protocol and Authentication Mechanism	SDK	Identity Type	Policy Type		
MQTT over mutual authentication (port 8883)	AWS IoT Device SDK	X.509 certificates	AWS IoT policy		
MQTT over WebSocket (port 443)	AWS Mobile SDK	Amazon Cognito, IAM, or federated identity	AWS IoT policy for Amazon Cognito identities IAM policy for other identities		
HTTP over server authentication (port 443)	AWS CLI	Amazon Cognito, IAM, or federated identity	AWS IoT policy for Amazon Cognito identities IAM policy for other identities		
HTTP over mutual authentication (port 8443)	No SDK support	X.509 certificates	AWS IoT policy		

AWS IoT Control Plane API and Policy Types

Protocol and Authentication Mechanism	SDK	Identity Type	Policy Type		
HTTP over server authentication (port 443)	AWS CLI	Amazon Cognito, IAM, or federated identity	AWS IoT policy for Amazon Cognito identities IAM policy for other identities		

AWS IoT policies are attached to X.509 certificates or Amazon Cognito identities. IAM policies are attached to an IAM user, group, or role. If you use the AWS IoT console or the AWS IoT CLI to attach the policy (to a certificate or Amazon Cognito Identity), you use an AWS IoT policy. Otherwise, you use an IAM policy.

Policy-based authorization is a powerful tool. It gives you complete control over what a device, user, or application can do in AWS IoT. For example, consider a device connecting to AWS IoT with a certificate. You can allow the device to access all MQTT topics, or you can restrict its access to a single topic. In another example, consider a user typing CLI commands at the command line. By using a policy, you can allow or deny access to any command or AWS IoT resource for the user. You can also control an application's access to AWS IoT resources.

AWS IoT Policies

AWS IoT policies are JSON documents. They follow the same conventions as IAM policies. AWS IoT supports named policies so many identities can reference the same policy document. Named policies are versioned so they can be easily rolled back.

AWS IoT defines a set of policy actions that describe the operations and resources to which you can grant or deny access. For example:

- `iot:Connect` represents permission to connect to the AWS IoT message broker.
- `iot:Subscribe` represents permission to subscribe to an MQTT topic or topic filter.
- `iot:GetThingShadow` represents permission to get a thing shadow.

AWS IoT policies allow you to control access to the AWS IoT data plane. The AWS IoT data plane consists of operations that allow you to connect to the AWS IoT message broker, send and receive MQTT messages, and get or update thing shadows. For more information, see AWS IoT Policy Actions (p. 122).

An AWS IoT policy is a JSON document that contains one or more policy statements. Each statement contains an `Effect`, an `Action`, and a `Resource`. The `Effect` specifies whether the action will be allowed or denied. The `Action` specifies the action the policy is allowing or denying. The `Resource` specifies the resource or resources on which the action is allowed or denied. The following policy grants all devices permission to connect to the AWS IoT message broker, but restricts the device to publishing on a specific MQTT topic:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action":["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/foo/bar"]
    },
    {
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": ["*"]
        }]
}
```

AWS IoT Policy Actions

The following policy actions are defined by AWS IoT:

MQTT Policy Actions

iot:Connect

Represents the permission to connect to the AWS IoT message broker. The `iot:Connect` permission is checked every time a `CONNECT` request is sent to the broker. The message broker does not allow two clients with the same client ID to stay connected at the same time. After the second client connects, the broker detects this case and disconnects one of the clients. The `iot:Connect` permission can be used to ensure only authorized clients can connect using a specific client ID.

iot:Publish

Represents the permission to publish on an MQTT topic. This permission is checked every time a PUBLISH request is sent to the broker. This can be used to allow clients to publish to specific topic patterns.

Note
You must also grant `iot:Connect` permission to grant `iot:Publish` permission.

iot:Receive

Represents the permission to receive a message from AWS IoT. The `iot:Receive` permission is checked every time a message is delivered to a client. Because this permission is checked on every delivery, it can be used to revoke permissions to clients that are currently subscribed to a topic.

iot:Subscribe

Represents the permission to subscribe to a topic filter. This permission is checked every time a SUBSCRIBE request is sent to the broker. This can be used to allow clients to subscribe to topics that match specific topic patterns.

Note
You must also grant `iot:Connect` permission to grant `iot:Subscribe` permission.

Thing Shadow Policy Actions

iot:DeleteThingShadow

Represents the permission to delete a thing shadow. The `iot:DeleteThingShadow` permission is checked every time a request is made to delete the thing shadow document.

iot:GetThingShadow

Represents the permission to retrieve a thing shadow. The `iot:GetThingShadow` permission is checked every time a request is made to retrieve a thing shadow document.

iot:UpdateThingShadow

Represents the permission to update a thing shadow. The `iot:UpdateThingShadow` permission is checked every time a request is made to update the state of a thing shadow document.

Action Resources

To specify a resource for an AWS IoT policy action, you must use the ARN of the resource. All resource ARNs are of the following form:

```
arn:aws:iot:region:AWS account ID:resource type/resource name
```

The following table shows the resource to specify for each action type:

Action	Resource
iot:DeleteThingShadow	A thing ARN - arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:Connect	A client ID ARN - arn:aws:iot:us-east1:123456789012:client/myClientId
iot:Publish	A topic ARN - arn:aws:iot:us-east-1:123456789012:topic/myTopicName

Action	Resource
iot:Subscribe	A topic filter ARN - arn:aws:iot:us-east-1:123456789012:topicfilter/myTopicFilter
iot:Receive	A topic ARN - arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iot:UpdateThingShadow	A thing ARN - arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:GetThingShadow	A thing ARN - arn:aws:iot:us-east-1:123456789012:thing/thingOne

AWS IoT Policy Variables

AWS IoT defines policy variables that can be used in AWS IoT policies within the resource or condition block. When a policy is evaluated, the policy variables are replaced by actual values. For example, if a device connected to the AWS IoT message broker with a client ID of "100-234-3456", the `iot:ClientId` policy variable would be replaced in the policy document by "100-234-3456". For more information about policy variables, see IAM Policy Variables and Multi-Value Conditions.

Basic Policy Variables

AWS IoT defines the following basic policy variables:

- `iot:ClientId`: The client ID used to connect to the AWS IoT message broker.
- `aws:SourceIp`: The IP address of the client connected to the AWS IoT message broker.

The following AWS IoT policy shows the use of policy variables:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": [
            "arn:aws:iot:us-east-1:123451234510:client/${iot:ClientId}"
        ]
    },
    {
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": [
            "arn:aws:iot:us-east-1:123451234510:topic/foo/bar/${iot:ClientId}"
        ]
    }]
}
```

In these examples, `${iot:ClientId}` is replaced by the ID of the client connected to the AWS IoT message broker when the policy is evaluated. When you use policy variables like `${iot:ClientId}`, you can inadvertently open access to unintended topics. For example, if you use a policy that uses `${iot:ClientId}` to specify a topic filter:

```
{
    "Effect": "Allow",
    "Action": ["iot:Subscribe"],
```

```
    "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/${iot:ClientId}/bar"
    ]
}
```

A client can connect using + as the client ID. This would allow the user to subscribe to any topic matching the topic filter `foo/+/bar`. To protect against such security gaps, use the `iot:Connect` policy action to control which client IDs are able to connect. For example, this policy allows only clients whose client ID is `clientid1` to connect:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:client/clientid1"
        ]
    }]
}
```

X.509 Certificate Policy Variables

X.509 certificate policy variables allow you to write AWS IoT policies that grant permissions based on X.509 certificate attributes. The following sections describe how you can use these certificate policy variables.

Issuer Attributes

The following AWS IoT policy variables allow you to allow or deny permissions based on certificate attributes set by the certificate issuer.

- `iot:Certificate.Issuer.DistinguishedNameQualifier`
- `iot:Certificate.Issuer.Country`
- `iot:Certificate.Issuer.Organization`
- `iot:Certificate.Issuer.OrganizationalUnit`
- `iot:Certificate.Issuer.State`
- `iot:Certificate.Issuer.CommonName`
- `iot:Certificate.Issuer.SerialNumber`
- `iot:Certificate.Issuer.Title`
- `iot:Certificate.Issuer.Surname`
- `iot:Certificate.Issuer.GivenName`
- `iot:Certificate.Issuer.Initials`
- `iot:Certificate.Issuer.Pseudonym`
- `iot:Certificate.Issuer.GenerationQualifier`

Subject Attributes

The following AWS IoT policy variables allow you to grant or deny permissions based on certificate subject attributes set by the certificate issuer.

- `iot:Certificate.Subject.DistinguishedNameQualifier`
- `iot:Certificate.Subject.Country`
- `iot:Certificate.Subject.Organization`
- `iot:Certificate.Subject.OrganizationalUnit`

- iot:Certificate.Subject.State
- iot:Certificate.Subject.CommonName
- iot:Certificate.Subject.SerialNumber
- iot:Certificate.Subject.Title
- iot:Certificate.Subject.Surname
- iot:Certificate.Subject.GivenName
- iot:Certificate.Subject.Initials
- iot:Certificate.Subject.Pseudonym
- iot:Certificate.Subject.GenerationQualifier

X.509 certificates allow these attributes to contain one or more values. By default, the policy variables for each multi-value attribute return the first value. For example, the Certificate.Subject.Country attribute might contain a list of country names. When evaluated in a policy, iot:Certificate.Subject.Country is replaced by the first country name. You can request a specific attribute value using a zero-based index. For example, iot:Certificate.Subject.Country#1 is replaced by the second country name in the Certificate.Subject.Country attribute. If you specify an attribute value that does not exist (for example, if you ask for a third value when there are only two values assigned to the attribute), no substitution is made and authorization fails. You can use the .List suffix on the policy variable name to specify all values of the attribute. The following example policy allows any client to connect to AWS IoT, but restricts publishing rights to those clients with certificates whose Certificate.Subject.Organization attribute is set to "Example Corp" or "AnyCompany". This is done through the use of a "Condition" attribute that specifies a condition for the preceding action. The condition in this case is that the Certificate.Subject.Organization attribute of the certificate must include one of the listed values.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Effect":"Allow",
            "Action":[
                "iot:Connect"
            ],
            "Resource":[
                "*"
            ]
        },
        {
            "Effect":"Allow",
            "Action":[
                "iot:Publish"
            ],
            "Resource":[
                "*"
            ],
            "Condition":{
                "ForAllValues:StringEquals":{
                    "iot:Certificate.Subject.Organization.List":[
                        "Example Corp",
                        "AnyCompany"
                    ]
                }
            }
        }
    ]
}
```

Issuer Alternate Name Attributes

The following AWS IoT policy variables allow you to grant or deny permissions based on issuer alternate name attributes set by the certificate issuer.

- `iot:Certificate.Issuer.AlternativeName.RFC822Name`
- `iot:Certificate.Issuer.AlternativeName.DNSName`
- `iot:Certificate.Issuer.AlternativeName.DirectoryName`
- `iot:Certificate.Issuer.AlternativeName.UniformResourceIdentifier`
- `iot:Certificate.Issuer.AlternativeName.IPAddress`

Subject Alternate Name Attributes

The following AWS IoT policy variables allow you to grant or deny permissions based on subject alternate name attributes set by the certificate issuer.

- `iot:Certificate.Subject.AlternativeName.RFC822Name`
- `iot:Certificate.Subject.AlternativeName.DNSName`
- `iot:Certificate.Subject.AlternativeName.DirectoryName`
- `iot:Certificate.Subject.AlternativeName.UniformResourceIdentifier`
- `iot:Certificate.Subject.AlternativeName.IPAddress`

Other Attributes

You can use `iot:Certificate.SerialNumber` to allow or deny access to AWS IoT resources based on the serial number of a certificate. The `iot:Certificate.AvailableKeys` policy variable contains the name of all certificate policy variables that contain values.

X.509 Certificate Policy Variable Limitations

The following limitations apply to X.509 certificate policy variables:

Wildcards

: If wildcard characters are present in certificate attributes, the policy variable is not replaced by the certificate attribute value, leaving the `${policy-variable}` text in the policy document. This might cause authorization failure.

Array fields

: Certificate attributes that contain arrays are limited to five items. Additional items are ignored.

String length

: All string values are limited to 1024 characters. If a certificate attribute contains a string longer than 1024 characters, the policy variable is not replaced by the certificate attribute value, leaving the `${policy-variable}` in the policy document. This might cause authorization failure.

Thing Policy Variables

Thing policy variables allow you to write AWS IoT policies that grant or deny permissions based on thing properties like thing names, thing types, and thing attribute values. The thing name is obtained from the client ID in the MQTT `Connect` message sent when a thing connects to AWS IoT. The thing policy variables are replaced when a thing connects to AWS IoT over MQTT using TLS mutual authentication or MQTT over the WebSocket protocol using authenticated Amazon Cognito identities. Thing policy variables are also replaced when a certificate or authenticated Amazon Cognito identity is attached to a

thing. You can use the AttachThingPrincipal API to attach certificates and authenticated Amazon Cognito identities to a thing.

The following thing policy variables are available:

- `iot:Connection.Thing.ThingName`
- `iot:Connection.Thing.ThingTypeName`
- `iot:Connection.Thing.Attributes[`*`attributeName`*`]`
- `iot:Connection.Thing.IsAttached`

iot:Connection.Thing.ThingName

This resolves to the name of the thing for which the policy is being evaluated. The thing name is set to the client ID of the MQTT/WebSocket connection. This policy variable is available only when connecting over MQTT or MQTT over the WebSocket protocol.

iot:Connection.Thing.ThingTypeName

This resolves to the thing type associated with the thing for which the policy is being evaluated. The thing name is set to the client ID of the MQTT/WebSocket connection. The thing type name is obtained by a call to the `DescribeThing` API. This policy variable is available only when connecting over MQTT or MQTT over the WebSocket protocol.

iot:Connection.Thing.Attributes[*attributeName*]

This resolves to the value of the specified attribute associated with the thing for which the policy is being evaluated. A thing can have up to 50 attributes. Each attribute is available as a policy variable: `iot:Connection.Thing.Attributes[`*`attributeName`*`]` where *`attributeName`* is the name of the attribute. The thing name is set to the client ID of the MQTT/WebSocket connection. This policy variable is only available when connecting over MQTT or MQTT over the WebSocket protocol.

iot:Connection.Thing.IsAttached

This resolves to `true` if the thing for which the policy is being evaluated has a certificate or Amazon Cognito identity attached.

Example Policies

AWS IoT policies are specified in a JSON document. These are the components of an AWS IoT policy:

Version

Must be set to `"2012-10-17"`.

Effect

Must be set to `"Allow"` or `"Deny"`.

Action

Must be set to "iot:*`operation-name`*" where *`operation-name`* is one of the following:

`"iot:Connect"`: Connect to AWS IoT

`"iot:Receive"`: Receive messages from AWS IoT

`"iot:Publish"`: MQTT publish.

`"iot:Subscribe"`: MQTT subscribe.

`"iot:UpdateThingShadow"`: Update a thing shadow.

`"iot:GetThingShadow"`:Retrieve a thing shadow.

`"iot:DeleteThingShadow"`:Delete a thing shadow.

Resource

Must be set to one of the following:

Client - arn:aws:iot:*region*:*account-id*:client/*client-id*

Topic ARN - arn:aws:iot:*region*:*account-id*:topic/*topic-name*

Topic filter ARN - arn:aws:iot:*region*:*account-id*:topicfilter/*topic-filter*

Connect Policy Examples

The following policy allows a set of client IDs to connect:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientid1",
                "arn:aws:iot:us-east-1:123456789012:client/clientid2",
                "arn:aws:iot:us-east-1:123456789012:client/clientid3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

The following policy prevents a set of client IDs from connecting:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientid1",
                "arn:aws:iot:us-east-1:123456789012:client/clientid2"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

The following policy allows the certificate holder using any client ID to subscribe to topic filter `foo/*`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/*"
            ]
        }
    ]
}
```

Publish/Subscribe Policy Examples

The policy you use depends on how you are connecting to AWS IoT. You can connect to AWS IoT using an MQTT client, HTTP, or WebSocket. When you connect with an MQTT client, you are authenticating with an X.509 certificate. When you connect over HTTP or the WebSocket protocol, you are authenticating with Signature Version 4 and Amazon Cognito.

Policies for MQTT Clients

When you specify topic filters in AWS IoT policies for MQTT clients, MQTT wildcard characters "+" and "#" are treated as literal characters. Their use might result in unexpected behavior. For example, the following policy allows a client to subscribe to the topic filter `foo/+/bar` only:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/+/bar"
            ]
        }
    ]
}
```

Note
The MQTT wildcard character '+' is not treated as a wildcard within a policy. Attempts to subscribe to topic filters that match the pattern `foo/+/bar` like `foo/baz/bar` or `foo/goo/bar` fails and causes the client to disconnect.

You can use "*" as a wildcard in the resource attribute of the policy. For example, the following policy allows the certificate holder to publish to all topics and subscribe to all topic filters in the AWS account:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

The following policy allows the certificate holder to publish to all topics in the AWS account:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

You can also use the "*" wildcard at the end of a topic filter. For example, the following policy allows the certificate holder to subscribe to a topic filter matching the pattern `foo/bar/*`:

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

```
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/bar/*"
            ]
        }
    ]
}
```

The following policy allows the certificate holder to publish to the `foo/bar` and `foo/baz` topics:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/foo/bar",
                "arn:aws:iot:us-east-1:123456789012:topic/foo/baz"
            ]
        }
    ]
}
```

The following policy prevents the certificate holder from publishing to the `foo/bar` topic:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/foo/bar"
```

```
            ]
        }
    ]
}
```

The following policy allows the certificate holder to publish on topic `foo` and prevents the certificate holder from publishing to topic `bar`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/foo"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/bar"
            ]
        }
    ]
}
```

The following policy allows the certificate holder to subscribe to topic filter `foo/bar`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/bar"
            ]
```

```
        }
    ]
}
```

The following policy allows the certificate holder to publish on the `arn:aws:iot:us-east-1:123456789012:topic/iotmonitor/provisioning/8050373158915119971` topic and allows the certificate holder to subscribe to the topic filter `arn:aws:iot:us-east-1:123456789012:topicfilter/iotmonitor/provisioning/8050373158915119971`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/iotmonitor/
provisioning/8050373158915119971"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/iotmonitor/
provisioning/8050373158915119971"
            ]
        }
    ]
}
```

Policies for HTTP and WebSocket Clients

For the following operations, AWS IoT uses AWS IoT policies attached to Amazon Cognito identities (through the `AttachPolicy` API) to scope down the permissions attached to the Amazon Cognito identity pool with authenticated identities. That means an Amazon Cognito identity needs permission from the IAM role policy attached to the pool and the AWS IoT policy attached to the Amazon Cognito identity through the AWS IoT `AttachPolicy` API.

- `iot:Connect`
- `iot:Publish`
- `iot:Subscribe`
- `iot:Receive`
- `iot:GetThingShadow`
- `iot:UpdateThingShadow`
- `iot:DeleteThingShadow`

Note
For other AWS IoT operations or for unauthenticated identities, AWS IoT does not scope down the permissions attached to the Amazon Cognito identity pool role. For both authenticated and unauthenticated identities, this is the most permissive policy that we recommend attaching to the Amazon Cognito pool role.

To allow unauthenticated Amazon Cognito identities to publish messages over HTTP on any topic, attach the following policy to the Amazon Cognito identity pool role:

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iot:Connect",
            "iot:Publish",
            "iot:Subscribe",
            "iot:Receive",
            "iot:GetThingShadow",
            "iot:UpdateThingShadow",
            "iot:DeleteThingShadow"
        ],
        "Resource": ["*"]
    }]
}
```

To allow unauthenticated Amazon Cognito identities to publish MQTT messages over HTTP on any topic in your account, attach the following policy to the Amazon Cognito identity pool role:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": ["*"]
    }]
}
```

Note
This example is for illustration only. Unless your service absolutely requires it, we recommend the use of a more restrictive policy, one that does not allow unauthenticated Amazon Cognito identities to publish on any topic.

To allow unauthenticated Amazon Cognito identities to publish MQTT messages over HTTP on `topic1` in your account, attach the following policy to your Amazon Cognito identity pool role:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic1"]
    }]
}
```

For an authenticated Amazon Cognito identity to publish MQTT messages over HTTP on `topic1` in your AWS account, you must specify two policies, as outlined here. The first policy must be attached to an Amazon Cognito identity pool role. It allows identities from that pool to make a publish call. The second policy must be attached to an Amazon Cognito user using the AWS IoT AttachPolicy API. It allows the specified Amazon Cognito user access to the `topic1` topic.

Amazon Cognito identity pool policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [ "iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic1"]
    }]
}
```

Amazon Cognito user policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic1"]
    }]
}
```

Similarly, the following example policy allows the Amazon Cognito user to publish MQTT messages over HTTP on the `topic1` and `topic2` topics. Two policies are required. The first policy gives the Amazon Cognito identity pool role the ability to make the publish call. The second policy gives the Amazon Cognito user access to the `topic1` and `topic2` topics.

Amazon Cognito identity pool policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": ["*"]
    }]
}
```

Amazon Cognito user policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/topic1",
            "arn:aws:iot:us-east-1:123456789012:topic/topic2"
        ]
    }]
}
```

The following policies allow multiple Amazon Cognito users to publish to a topic. Two policies per Amazon Cognito identity are required. The first policy gives the Amazon Cognito identity pool role the ability to make the publish call. The second and third policies give the Amazon Cognito users access to the topics `topic1` and `topic2`, respectively.

Amazon Cognito identity pool policy:

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": ["*"]
    }]
}
```

Amazon Cognito user1 policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic1"]
    }]
}
```

Amazon Cognito user2 policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic2"]
    }]
}
```

Receive Policy Examples

The following policy prevents the certificate holder using any client ID from receiving messages from a topic:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/foo/restricted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

The following policy allows the certificate holder using any client ID to subscribe and receive messages on one topic:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [*]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/bar"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/foo/bar"
            ]
        }
    ]
}
```

Certificate Policy Examples

The following policy allows a device to publish on a topic whose name is equal to the `certificateId` of the certificate with which the device authenticated itself:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action":["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${iot:CertificateId}"]
    },
    {
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": ["*"]
    }]
}
```

The following policy allows a device to publish on a topic whose name is equal to the subject's common name field of the certificate with which the device authenticated itself:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action":["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Issuer.CommonName}"]
    },
    {
```

```
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": ["*"]
    }]
}
```

The following policy allows a device to publish on a topic that is prefixed with "admin/" when the certificate used to authenticate the device has its `Subject.CommonName.2` field set to "Administrator":

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": ["*"]
    },
    {
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
        "Condition": {
            "StringEquals": {
                "iot:Certificate.Subject.CommonName.2": "Administrator"
            }
        }
    }]
}
```

The following policy allows a device to publish on a topic that is prefixed with "admin/" when the certificate used to authenticate the device has any one of its `Subject.Common` fields set to "Administrator":

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": ["*"]
    },
    {
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
        "Condition": {
            "ForAnyValue:StringEquals": {
                "iot:Certificate.Subject.CommonName.List": "Administrator"
            }
        }
    }]
}
```

Thing Policy Examples

The following policy allows a thing to publish on a specific topic that contains the thing type name and thing name:

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":["iot:Publish"],
        "Resource":[
```

```
            "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingTypeName}/${iot:Connection.Thing.ThingName}"
        ]
    }]
}
```

The following policy allows the device to connect if the certificate used to authenticate with AWS IoT is attached to the thing for which the policy is being evaluated.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": ["*"],
        "Condition":{
            "Bool":{
                "iot:Connection.Thing.IsAttached":["true"]
            }
        }
    }]
}
```

The following policy allows a device to publish on a set of topics ("/foo/bar" and "/foo/baz") if:

- The thing associated with the device has an attribute called "Manufacturer" with a value of "foo", "bar", or "baz".
- The thing associated with the device exists in the thing registry and is attached to the certificate used to connect to AWS IoT.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/foo/bar",
            "arn:aws:iot:us-east-1:123456789012:topic/foo/baz"
        ],
        "Condition": {
            "ForAnyValue:StringLike": {
                "iot:Connection.Thing.Attributes[Manufacturer]": [
                    "foo",
                    "bar",
                    "baz"
                ]
            }
        }
    }]
}
```

The following policy allows a device to publish to a topic if:

- The topic is composed of the thing type name, a '/', and the thing name.
- The thing exists in the thing registry.
- The thing is attached to the certificate used to connect to AWS IoT.

```
{
```

```
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":["iot:Publish"],
        "Resource":[
            "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingTypeName}/${iot:Connection.Thing.ThingName}"
         ]
    }]
}
```

The following policy allows a device to publish only on its own thing shadow topic, if the thing exists in the thing registry.

```
{
    "Version": "2012-10-17",
    "Statement": [{
       "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/update"
        ]
    }]
}
```

IAM IoT Policies

AWS Identity and Access Management defines a policy action for each operation defined by AWS IoT, including control plane and data plane APIs.

AWS IoT API Permissions

The following table lists the AWS IoT API, the IAM permissions required, and the resource the API manipulates.

API	Required Permission (Policy Actions)	Resources
AcceptCertificateTransfer	iot:AcceptCertificateTransfer	arn:aws:iot:*region*:*account-id*:cert/*cert-id* **Note** The AWS account specified in the ARN must be the account to which the certificate is being transferred.
AddLoggingRole	iot:AddLoggingRole	none
AddThingToThingGroup	iot:AddThingToThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* arn:aws:iot:*region*:*account-id*:thing/*thing-name*
AssociateTargetsWithJob	iot:AssociateTargetsWithJob	none
AttachPolicy	iot:AttachPolicy	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* or

API	Required Permission (Policy Actions)	Resources
		arn:aws:iot:*region*:*account-id*:cert/*cert-id*
AttachPrincipalPolicy	iot:AttachPrincipalPolicy	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
AttachThingPrincipal	iot:AttachThingPrincipal	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
CancelCertificateTransfer	iot:CancelCertificateTransfer	arn:aws:iot:*region*:*account-id*:cert/*cert-id* **Note** The AWS account specified in the ARN must be the account to which the certificate is being transferred.
CancelJob	iot:CancelJob	arn:aws:iot:*region*:*account-id*:job/*job-id*
ClearDefaultAuthorizer	iot:ClearDefaultAuthorizer	*
CreateAuthorizer	iot:CreateAuthorizer	arn:aws:iot:*region*:*account-id*:authorizer/*authorizer-function-name*
CreateCertificateFromCsr	iot:CreateCertificateFromCsr	*
CreateJob	iot:CreateJob	arn:aws:iot:*region*:*account-id*:job/*job-id*
CreateKeysAndCertificate	iot:CreateKeysAndCertificate	*
CreateMessageSchema	iot:CreateMessageSchema	name
CreatePolicy	iot:CreatePolicy	*
CreatePolicyVersion	iot:CreatePolicyVersion	arn:aws:iot:*region*:*account-id*:policy/*policy-name* **Note** This must be an AWS IoT policy, not an IAM policy.
CreateRoleAlias	iot:CreateRoleAlias	(parameter: roleAlias) arn:aws:iot:*region*:*account-id*:rolealias/*role-alias-name*
CreateThing	iot:CreateThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
CreateThingGroup	iot:CreateThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* for group being created and for parent group, if used
CreateThingType	iot:CreateThingType	arn:aws:iot:*region*:*account-id*:thingtype/*thing-type-name*
CreateTopicRule	iot:CreateTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
DeleteAuthorizer	iot:DeleteAuthorizer	arn:aws:iot:*region*:*account-id*:authorizer/*authorizer-name*
DeleteCACertificate	iot:DeleteCACertificate	arn:aws:iot:*region*:*account-id*:cacert/*cert-id*
DeleteCertificate	iot:DeleteCertificate	arn:aws:iot:*region*:*account-id*:cert/*cert-id*

API	Required Permission (Policy Actions)	Resources
DeleteLoggingLevel	iot:DeleteLoggingLevel	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
DeleteLoggingRole	iot:DeleteLoggingRole	none
DeleteMessageSchema	iot:DeleteMessageSchema	none
DeletePolicy	iot:DeletePolicy	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
DeletePolicyVersion	iot:DeletePolicyVersion	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
DeleteRegistrationCode	iot:DeleteRegistrationCode	*
DeleteRoleAlias	iot:DeleteRoleAlias	arn:aws:iot:*region*:*account-id*:rolealias/*role-alias-name*
DeleteThing	iot:DeleteThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
DeleteThingGroup	iot:DeleteThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
DeleteThingType	iot:DeleteThingType	arn:aws:iot:*region*:*account-id*:thingtype/*thing-type-name*
DeleteTopicRule	iot:DeleteTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
DeleteV2LoggingLevel	iot:DeleteV2LoggingLevel	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
DeprecateThingType	iot:DeprecateThingType	arn:aws:iot:*region*:*account-id*:thingtype/*thing-type-name*
DescribeAuthorizer	iot:DescribeAuthorizer	arn:aws:iot:*region*:*account-id*:authorizer/*authorizer-function-name* (parameter: authorizerName) none
DescribeCACertificate	iot:DescribeCACertificate	arn:aws:iot:*region*:*account-id*:cacert/*cert-id*
DescribeCertificate	iot:DescribeCertificate	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
DescribeDefaultAuthorizer	iot:DescribeDefaultAuthorizer	none
DescribeEndpoint	iot:DescribeEndpoint	*
DescribeEventConfigurations	iot:DescribeEventConfigurations	none
DescribeIndex	iot:DescribeIndex	arn:aws:iot:*region*:*account-id*:index/*index-name*
DescribeJob	iot:DescribeJob	arn:aws:iot:*region*:*account-id*:job/*job-id*
DescribeJobExecution	iot:DescribeJobExecution	none
DescribeRoleAlias	iot:DescribeRoleAlias	arn:aws:iot:*region*:*account-id*:rolealias/*role-alias-name*
DescribeThing	iot:DescribeThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*

API	Required Permission (Policy Actions)	Resources
DescribeThingGroup	iot:DescribeThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
DescribeThingRegistrationTask	iot:DescribeThingRegistrationTask	arn:aws:iot:*region*:*account-id*:thingregistrationTask
DescribeThingType	iot:DescribeThingType	arn:aws:iot:*region*:*account-id*:thingtype/*thing-type-name*
DetachPolicy	iot:DetachPolicy	arn:aws:iot:*region*:*account-id*:cert/*cert-id* or arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
DetachPrincipalPolicy	iot:DetachPrincipalPolicy	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
DetachThingPrincipal	iot:DetachThingPrincipal	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
DisableTopicRule	iot:DisableTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
EnableTopicRule	iot:EnableTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
GetEffectivePolicies	iot:GetEffectivePolicies	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
GetIndexingConfiguration	iot:GetIndexingConfiguration	none
GetJobDocument	iot:GetJobDocument	arn:aws:iot:*region*:*account-id*:job/*job-id*
GetLoggingOptions	iot:GetLoggingOptions	*
GetLoggingOptionsV2	iot:GetLoggingOptionsV2	none
GetLoggingRole	iot:GetLoggingRole	none
GetMessageSchema	iot:GetMessageSchema	none
GetPolicy	iot:GetPolicy	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
GetPolicyVersion	iot:GetPolicyVersion	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
GetRegistrationCode	iot:GetRegistrationCode	*
GetTopicRule	iot:GetTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
GetV2LoggingOptions	iot:GetV2LoggingOptions	none
ListAttachedPolicies	iot:ListAttachedPolicies	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* or arn:aws:iot:*region*:*account-id*:cert/*cert-id*
ListAuthorizers	iot:ListAuthorizers	none
ListCACertificates	iot:ListCACertificates	*

API	Required Permission (Policy Actions)	Resources
ListCertificates	iot:ListCertificates	*
ListCertificatesByCA	iot:ListCertificatesByCA	*
ListIndices	iot:ListIndices	none
ListJobExecutionsForJob	iot:ListJobExecutionsForJob	none
ListJobExecutionsForThing	iot:ListJobExecutionsForThing	none
ListJobs	iot:ListJobs	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* if thingGroupName parameter used
ListLoggingLevels	iot:ListLoggingLevels	none
ListMessageSchemas	iot:ListMessageSchemas	none
ListOutgoingCertificates	iot:ListOutgoingCertificates	*
ListPolicies	iot:ListPolicies	*
ListPolicyPrincipals	iot:ListPolicyPrincipals	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
ListPolicyVersions	iot:ListPolicyVersions	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
ListPrincipalPolicies	iot:ListPrincipalPolicies	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
ListPrincipalThings	iot:ListPrincipalThings	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
ListRoleAliases	iot:ListRoleAliases	none
ListTargetsForPolicy	iot:ListTargetsForPolicy	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
ListThingGroups	iot:ListThingGroups	none
ListThingGroupsForThing	iot:ListThingGroupsForThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
ListThingPrincipals	iot:ListThingPrincipals	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
ListThingRegistrationTaskReports	iot:ListThingRegistrationTaskReports	none
ListThingRegistrationTasks	iot:ListThingRegistrationTasks	none
ListThingTypes	iot:ListThingTypes	*
ListThings	iot:ListThings	*
ListThingsInThingGroup	iot:ListThingsInThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
ListTopicRules	iot:ListTopicRules	*
ListV2LoggingLevels	iot:ListV2LoggingLevels	none
RegisterCACertificate	iot:RegisterCACertificate	*
RegisterCertificate	iot:RegisterCertificate	*

API	Required Permission (Policy Actions)	Resources
RegisterThing	iot:RegisterThing	none
RejectCertificateTransfer	iot:RejectCertificateTransfer	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
RemoveThingFromThingGroup	iot:RemoveThingFromThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* arn:aws:iot:*region*:*account-id*:thing/*thing-name*
ReplaceTopicRule	iot:ReplaceTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
SearchIndex	iot:SearchIndex	arn:aws:iot:*region*:*account-id*:index/*index-id*
SetDefaultAuthorizer	iot:SetDefaultAuthorizer	arn:aws:iot:*region*:*account-id*:authorizer/*authorizer-function-name*
SetDefaultPolicyVersion	iot:SetDefaultPolicyVersion	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
SetLoggingLevel	iot:SetLoggingLevel	none
SetLoggingOptions	iot:SetLoggingOptions	arn:aws:iot:*region*:*account-id*:role/*role-name*
SetLoggingOptionsV2	iot:SetLoggingOptionsV2	arn:aws:iot:*region*:*account-id*:role/*role-name*
SetV2LoggingLevel	iot:SetV2LoggingLevel	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
SetV2LoggingOptions	iot:SetV2LoggingOptions	arn:aws:iot:*region*:*account-id*:role/*role-name*
StartThingRegistrationTask	iot:StartThingRegistrationTask	none
StopThingRegistrationTask	iot:StopThingRegistrationTask	none
TestAuthorization	iot:TestAuthorization	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
TestInvokeAuthorizer	iot:TestInvokeAuthorizer	none
TransferCertificate	iot:TransferCertificate	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
UpdateAuthorizer	iot:UpdateAuthorizer	arn:aws:iot:*region*:*account-id*:authorizerfunction/*authorizer-function-name*
UpdateCACertificate	iot:UpdateCACertificate	arn:aws:iot:*region*:*account-id*:cacert/*cert-id*
UpdateCertificate	iot:UpdateCertificate	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
UpdateEventConfigurations	iot:UpdateEventConfigurations	none
UpdateIndexingConfiguration	iot:UpdateIndexingConfiguration	none
UpdateMessageSchema	iot:UpdateMessageSchema	none
UpdateRoleAlias	iot:UpdateRoleAlias	arn:aws:iot:*region*:*account-id*:rolealias/*role-alias-name*
UpdateThing	iot:UpdateThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*

API	Required Permission (Policy Actions)	Resources
UpdateThingGroup	iot:UpdateThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
UpdateThingGroupsForThing	iot:UpdateThingGroupsForThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*

IAM Policy Templates

AWS IoT provides a set of IAM policy templates you can either use as-is or as a starting point for creating custom IAM policies. These templates allow access to configuration and data operations. Configuration operations allow you to create things, certificates, policies, and rules. Data operations send data over MQTT or HTTP protocols. The following table describes these templates.

Policy Template	Description
AWSIotLogging	Allows the associated identity to configure CloudWatch logging. This policy is attached to your CloudWatch logging role.
AWSIoTConfigAccess	Allows the associated identity access to all AWS IoT configuration operations.
AWSIoTConfigReadOnlyAccess	Allows the associated identity to call read-only configuration operations.
AWSIoTDataAccess	Allows the associated identity full access to all AWS IoT data operations. Data operations send data over MQTT or HTTP protocols.
AWSIoTFullAccess	Allows the associated identity full access to all AWS IoT configuration and data operations.
AWSIoTRuleActions	Allows the associated identity access to all AWS services supported in AWS IoT rule actions.

Authorizing Direct Calls to AWS Services

Devices may use X.509 certificates to connect to AWS IoT using TLS mutual authentication protocols. Other AWS services do not support certificate-based authentication, but they can be called using AWS credentials in Signature Version 4 format. After authenticating a device with a certificate, AWS IoT can assume a role on behalf of the device and request temporary credentials from IAM. The device can then use that credential to call other AWS services.

Requests for temporary credentials can be made with an HTTP GET on port 443, for example:

```
https://<your-aws-account-id>.credentials.iot.region.amazonaws.com:443/role-aliases/<your-role-alias>/credentials
```

Note
To find your endpoint, use the `describe-endpoint` CLI command specifying `iot:CredentialProvider` as the endpoint type.

To make sure your device is communicating with AWS IoT (and not a service impersonating it), copy the Amazon Root CA 1 for RSA, and the VeriSign Class 3 Public Primary G5 root CA certificate to your device.

When making a request for temporary credentials, you can optionally provide a thing name in a request header called `x-amzn-iot-thingname`. In order to use `thingName` in a request header you must attach the thing to a device certificate using the AttachThingPrincipal API. When a device requests credentials by passing the `thingName`, AWS IoT checks that the `thingName` is attached to the certificate presented during the TLS handshake and will not provide temporary credentials unless it is.

Passing the thing name in the request allows you to use `thingName` and `thingType` as policy variables in the role's access policy for fine-grained access. For more information, see AWS IoT Policy Variables (p. 124). You cannot use thing variables in policies unless a thing name is passed in the request header.

The policy attached to the device certificate must grant the device permission to assume the role. You do this by granting permission for the `iot:AssumeRoleWithCertificate` action on the ARN of the role alias, for example `arn:aws:iot:`*`<your-region>`*`:`*`<your-aws-account-id>`*`:rolealias/`*`<role-alias-name>`*

You grant privileges to the temporary credentials by creating an IAM role and attaching policies to it. You can have fine-grained control over the privileges granted to this role by using policy variables `thingName`, `thingType` and `certificateId`. For more information, see AWS IoT Policy Variables (p. 124).

The device which is going to make direct calls to AWS Services must know what role ARN to use when connecting to AWS IoT. But hard-coding the role ARN is not a good solution because you would have to update the device anytime the role ARN changes. A better solution is to create a role alias that points to the role ARN and use that on your device. If the role ARN changes, you can update the role alias and no change is required on the device. Role aliases are created using the `CreateRoleAlias` API. This API takes the following parameters:

`credentialDurationInSeconds`

How long (in seconds) the credential is valid.

`roleAlias`

An arbitrary string identifying the role alias. Must be 1-128 characters and must include only A-Za-z0-9=,@- characters.

`roleArn`

The ARN of the role to which the role alias refers.

Note that the entity which performs the `CreateRoleAlias` must have sufficient privileges of its own to do so. Specifically, it must have an attached policy that allows the `iam:PassRole` action on the ARN of the created IAM role which is to be aliased.

You can pass a thing name in a request header when requesting temporary credentials. If the thing name is present, you can use thing policy variables to scope-down the credential returned by AWS IoT.

ThingName is an optional request parameter, which can be passed through an HTTP request header called `x-amzn-iot-thingname`.

Requests to AWS IoT for temporary credentials are made to port 443 over HTTP with TLS mutual authentication. This request must be an HTTP GET request. The URL is similar to:

```
https://<your-iot-endpoint>.iot.<your-aws-region>.amazonaws.com:443/role-
aliases/<roleAlias>/credentials
```

Cross Account Access

AWS IoT allows you to enable a principal to publish or subscribe to a topic that is defined in an AWS account not owned by the principal. You configure cross account access by creating an IAM policy and IAM role and then attaching the policy to the role.

First, create an IAM policy just like you would for other users and certificates in your AWS account. For example, the following policy grants permissions to connect and publish to the `/foo/bar` topic.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iot:Connect"
         ],
         "Resource": [
             "*"
         ]
     },
     {
         "Effect": "Allow",
         "Action": [
             "iot:Publish"
         ],
         "Resource": [
             "arn:aws:iot:us-east-1:123456789012:topic/foo/bar"
         ]
     }]
}
```

Next, follow the steps in Creating a Role for an IAM User. Enter the AWS account ID of the AWS account with which you want to share access. Then, in the final step, attach the policy you just created to the role. If, at a later time, you need to modify the AWS account ID to which you are granting access, you can use the following trust policy format to do so.

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect": "Allow",
        "Principal": {
             "AWS": "arn:aws:iam:us-east-1:111111111111:user/MyUser"
        },
        "Action": "sts:AssumeRole"
    }]
}
```

Transport Security

The AWS IoT message broker and Thing Shadows service encrypt all communication with TLS. TLS is used to ensure the confidentiality of the application protocols (MQTT, HTTP) supported by AWS IoT. TLS is available in a number of programming languages and operating systems.

For MQTT, TLS encrypts the connection between the device and the broker. TLS client authentication is used by AWS IoT to identify devices. For HTTP, TLS encrypts the connection between the device and the broker. Authentication is delegated to AWS Signature Version 4.

TLS Cipher Suite Support

AWS IoT supports the following cipher suites:

- ECDHE-ECDSA-AES128-GCM-SHA256 (recommended)
- ECDHE-RSA-AES128-GCM-SHA256 (recommended)
- ECDHE-ECDSA-AES128-SHA256
- ECDHE-RSA-AES128-SHA256
- ECDHE-ECDSA-AES128-SHA
- ECDHE-RSA-AES128-SHA
- ECDHE-ECDSA-AES256-GCM-SHA384
- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-ECDSA-AES256-SHA384
- ECDHE-RSA-AES256-SHA384
- ECDHE-RSA-AES256-SHA
- ECDHE-ECDSA-AES256-SHA
- AES128-GCM-SHA256
- AES128-SHA256
- AES128-SHA
- AES256-GCM-SHA384
- AES256-SHA256
- AES256-SHA

Message Broker for AWS IoT

The AWS IoT message broker is a publish/subscribe broker service that enables the sending and receiving of messages to and from AWS IoT. When communicating with AWS IoT, a client sends a message addressed to a topic like `Sensor/temp/room1`. The message broker, in turn, sends the message to all clients that have registered to receive messages for that topic. The act of sending the message is referred to as *publishing*. The act of registering to receive messages for a topic filter is referred to as *subscribing*.

The topic namespace is isolated for each AWS account and region pair. For example, the `Sensor/temp/room1` topic for an AWS account is independent from the `Sensor/temp/room1` topic for another AWS account. This is true of regions, too. The `Sensor/temp/room1` topic in the same AWS account in us-east-1 is independent from the same topic in us-east-2. AWS IoT does not support sending and receiving messages across AWS accounts and regions.

The message broker maintains a list of all client sessions and the subscriptions for each session. When a message is published on a topic, the broker checks for sessions with subscriptions that map to the topic. The broker then forwards the publish message to all sessions that have a currently connected client.

Protocols

The message broker supports the use of the MQTT protocol to publish and subscribe and the HTTPS protocol to publish. Both protocols are supported through IP version 4 and IP version 6. The message broker also supports MQTT over the WebSocket protocol.

Protocol/Port Mappings

The following table shows each protocol supported by AWS IoT, the authentication method, and port used for each protocol.

Protocol, Authentication, and Port Mappings

Protocol	Authentication	Port
MQTT	Client Certificate	8883
HTTP	Client Certificate	8443
HTTP	SigV4	443
MQTT + WebSocket	SigV4	443

MQTT

MQTT is a widely adopted lightweight messaging protocol designed for constrained devices. For more information, see MQTT.

Although the AWS IoT message broker implementation is based on MQTT version 3.1.1, it deviates from the specification as follows:

- In AWS IoT, subscribing to a topic with Quality of Service (QoS) 0 means a message will be delivered zero or more times. A message might be delivered more than once. Messages delivered more than once might be sent with a different packet ID. In these cases, the DUP flag is not set.
- AWS IoT does not support publishing and subscribing with QoS 2. The AWS IoT message broker does not send a PUBACK or SUBACK when QoS 2 is requested.
- The QoS levels for publishing and subscribing to a topic have no relation to each other. One client can subscribe to a topic using QoS 1 while another client can publish to the same topic using QoS 0.
- When responding to a connection request, the message broker sends a CONNACK message. This message contains a flag to indicate if the connection is resuming a previous session. The value of this flag might be incorrect if two MQTT clients connect with the same client ID simultaneously.
- When a client subscribes to a topic, there might be a delay between the time the message broker sends a SUBACK and the time the client starts receiving new matching messages.
- The MQTT specification provides a provision for the publisher to request that the broker retain the last message sent to a topic and send it to all future topic subscribers. AWS IoT does not support retained messages. If a request is made to retain messages, the connection is disconnected.
- The message broker uses the client ID to identify each client. The client ID is passed in from the client to the message broker as part of the MQTT payload. Two clients with the same client ID are not allowed to be connected concurrently to the message broker. When a client connects to the message broker using a client ID that another client is using, a CONNACK message will be sent to both clients and the currently connected client will be disconnected.
- The message broker does not support persistent sessions (connections made with the `cleanSession` flag set to `false`. The AWS IoT message broker assumes all sessions are clean sessions and messages are not stored across sessions. If an MQTT client attempts to connect to the AWS IoT message broker with the `cleanSession` set to `false`, the client will be disconnected.
- On rare occasions, the message broker might resend the same logical PUBLISH message with a different packet ID.
- The message broker does not guarantee the order in which messages and ACK are received.

HTTP

The message broker supports clients connecting with the HTTP protocol using a REST API. Clients can publish by sending a POST message to *<AWS IoT Endpoint>*/*<url_encoded_topic_name>*?qos=1".

For example, you can use curl to emulate a button press. If you followed the tutorial in Getting Started with AWS IoT (p. 5), rather than using the AWS IoT MQTT client to publish a message as you did in AWS IoT MQTT Client (p. 29), use something like the following command:

```
curl --tlsv1.2 --cacert root-CA.crt --cert 4b7828d2e5-certificate.pem.crt --key 4b7828d2e5-
private.pem.key -X POST -d "{ \"serialNumber\": \"G030JF053216F1BS\", \"clickType
\": \"SINGLE\", \"batteryVoltage\": \"2000mV\" }" "https://a1pn10j0v8htvw.iot.us-
east-1.amazonaws.com:8443/topics/iotbutton/virtualButton?qos=1"
```

--tlsv1.2

Use TLSv1.2 (SSL). curl must be installed with OpenSSL and you must use version 1.2 of TLS.

--cacert <filename>

The filename of the CA certificate to verify the peer.

--cert <filename>

The client certificate filename.

--key <filename>

The private key filename.

-X POST

The type of request, in this case, POST.

-d <data>

The HTTP POST data you want to publish. In this case, we emulate the data sent by a single button press.

"https://..."

The URL. In this case the REST API endpoint for the thing. (To find the endpoint for a thing, from the AWS IoT console choose **Registry** to expand your choices. Choose **Things**, choose the thing, and then choose **Interact**.) After the endpoint add the port (:8443) followed by the topic and, finally, specify the quality of service in a query string (?qos=1).

MQTT Over the WebSocket Protocol

AWS IoT supports MQTT over the WebSocket protocol to enable browser-based and remote applications to send and receive data from AWS IoT-connected devices using AWS credentials. AWS credentials are specified using AWS Signature Version 4. WebSocket support is available on TCP port 443, which allows messages to pass through most firewalls and web proxies.

A WebSocket connection is initiated on a client by sending an HTTP GET request. The URL you use is of the following form:

```
wss://<endpoint>.iot.<region>.amazonaws.com/mqtt
```

wss

Specifies the WebSocket protocol.

endpoint

Your AWS account-specific AWS IoT endpoint. You can use the AWS IoT CLI describe-endpoint command to find this endpoint.

region

The AWS region of your AWS account.

mqtt

Specifies you will be sending MQTT messages over the WebSocket protocol.

When the server responds, the client sends an upgrade request to indicate to the server it will communicate using the WebSocket protocol. After the server acknowledges the upgrade request, all communication is performed using the WebSocket protocol. The WebSocket implementation you use acts as a transport protocol. The data you send over the WebSocket protocol are MQTT messages.

Using the WebSocket Protocol in a Web Application

The WebSocket implementation provided by most web browsers does not allow the modification of HTTP headers, so you must add the Signature Version 4 information to the query string. For more information, see Adding Signing Information to the Query String.

The following JavaScript defines some utility functions used in generating a Signature Version 4 request.

```
  /**
   * utilities to do sigv4
   * @class SigV4Utils
   */
  function SigV4Utils() {}

SigV4Utils.getSignatureKey = function (key, date, region, service) {
    var kDate = AWS.util.crypto.hmac('AWS4' + key, date, 'buffer');
    var kRegion = AWS.util.crypto.hmac(kDate, region, 'buffer');
    var kService = AWS.util.crypto.hmac(kRegion, service, 'buffer');
    var kCredentials = AWS.util.crypto.hmac(kService, 'aws4_request', 'buffer');
    return kCredentials;
};

SigV4Utils.getSignedUrl = function(host, region, credentials) {
    var datetime = AWS.util.date.iso8601(new Date()).replace(/[:\-]|\.\d{3}/g, '');
    var date = datetime.substr(0, 8);

    var method = 'GET';
    var protocol = 'wss';
    var uri = '/mqtt';
    var service = 'iotdevicegateway';
    var algorithm = 'AWS4-HMAC-SHA256';

    var credentialScope = date + '/' + region + '/' + service + '/' + 'aws4_request';
    var canonicalQuerystring = 'X-Amz-Algorithm=' + algorithm;
    canonicalQuerystring += '&X-Amz-Credential=' +
 encodeURIComponent(credentials.accessKeyId + '/' + credentialScope);
    canonicalQuerystring += '&X-Amz-Date=' + datetime;
    canonicalQuerystring += '&X-Amz-SignedHeaders=host';

    var canonicalHeaders = 'host:' + host + '\n';
    var payloadHash = AWS.util.crypto.sha256('', 'hex')
    var canonicalRequest = method + '\n' + uri + '\n' + canonicalQuerystring + '\n' +
 canonicalHeaders + '\nhost\n' + payloadHash;

    var stringToSign = algorithm + '\n' + datetime + '\n' + credentialScope + '\n' +
 AWS.util.crypto.sha256(canonicalRequest, 'hex');
    var signingKey = SigV4Utils.getSignatureKey(credentials.secretAccessKey, date, region,
 service);
    var signature = AWS.util.crypto.hmac(signingKey, stringToSign, 'hex');

    canonicalQuerystring += '&X-Amz-Signature=' + signature;
    if (credentials.sessionToken) {
        canonicalQuerystring += '&X-Amz-Security-Token=' +
 encodeURIComponent(credentials.sessionToken);
    }

    var requestUrl = protocol + '://' + host + uri + '?' + canonicalQuerystring;
    return requestUrl;
};
```

To create a Signature Version 4 request

1. Create a canonical request for Signature Version 4.

   The following JavaScript code creates a canonical request:

   ```
   var datetime = AWS.util.date.iso8601(new Date()).replace(/[:\-]|\.\d{3}/g, '');
   var date = datetime.substr(0, 8);
   ```

```
var method = 'GET';
var protocol = 'wss';
var uri = '/mqtt';
var service = 'iotdevicegateway';
var algorithm = 'AWS4-HMAC-SHA256';

var credentialScope = date + '/' + region + '/' + service + '/' + 'aws4_request';
var canonicalQuerystring = 'X-Amz-Algorithm=' + algorithm;
canonicalQuerystring += '&X-Amz-Credential=' +
 encodeURIComponent(credentials.accessKeyId + '/' + credentialScope);
canonicalQuerystring += '&X-Amz-Date=' + datetime;
canonicalQuerystring += '&X-Amz-SignedHeaders=host';

var canonicalHeaders = 'host:' + host + '\n';
var payloadHash = AWS.util.crypto.sha256('', 'hex')
var canonicalRequest = method + '\n' + uri + '\n' + canonicalQuerystring + '\n' +
 canonicalHeaders + '\nhost\n' + payloadHash;
```

2. Create a string to sign, generate a signing key, and sign the string.

   Take the canonical URL you created in the previous step and assemble it into a string to sign. You do this by creating a string composed of the hashing algorithm, the date, the credential scope, and the SHA of the canonical request. Next, generate the signing key and sign the string, as shown in the following JavaScript code.

   ```
   var stringToSign = algorithm + '\n' + datetime + '\n' + credentialScope + '\n' +
    AWS.util.crypto.sha256(canonicalRequest, 'hex');
   var signingKey = SigV4Utils.getSignatureKey(credentials.secretAccessKey, date, region,
    service);
   var signature = AWS.util.crypto.hmac(signingKey, stringToSign, 'hex');
   ```

3. Add the signing information to the request.

   The following JavaScript code shows how to add the signing information to the query string.

   ```
   canonicalQuerystring += '&X-Amz-Signature=' + signature;
   ```

4. If you have session credentials (from an STS server, AssumeRole, or Amazon Cognito), append the session token to the end of the URL string after signing:

   ```
   canonicalQuerystring += '&X-Amz-Security-Token=' +
    encodeURIComponent(credentials.sessionToken);
   ```

5. Prepend the protocol, host, and URI to the canonicalQuerystring:

   ```
   var requestUrl = protocol + '://' + host + uri + '?' + canonicalQuerystring;
   ```

6. Open the WebSocket.

   The following JavaScript code shows how to create a Paho MQTT client and call CONNECT to AWS IoT. The `endpoint` argument is your AWS account-specific endpoint. The `clientId` is a text identifier that is unique among all clients simultaneously connected in your AWS account.

   ```
   var client = new Paho.MQTT.Client(requestUrl, clientId);
   ```

```
var connectOptions = {
    onSuccess: function(){
        // connect succeeded
    },
    useSSL: true,
    timeout: 3,
    mqttVersion: 4,
    onFailure: function() {
        // connect failed
    }
};
client.connect(connectOptions);
```

Using the WebSocket Protocol in a Mobile Application

We recommend using one of the AWS IoT Device SDKs to connect your device to AWS IoT when making a WebSocket connection. The following AWS IoT Device SDKs support WebSocket-based MQTT connections to AWS IoT:

- Node.js
- iOS
- Android

For a reference implementation for connecting a web application to AWS IoT using MQTT over the WebSocket protocol, see AWS Labs WebSocket sample.

If you are using a programming or scripting language that is not currently supported, any existing WebSocket library can be used as long as the initial WebSocket upgrade request (HTTP POST) is signed using AWS Signature Version 4. Some MQTT clients, such as Eclipse Paho for JavaScript, support the WebSocket protocol natively.

Topics

The message broker uses topics to route messages from publishing clients to subscribing clients. The forward slash (/) is used to separate topic hierarchy. The following table lists the wildcards that can be used in the topic filter when you subscribe.

Topic Wildcards

Wildcard	Description
#	Must be the last character in the topic to which you are subscribing. Works as a wildcard by matching the current tree and all subtrees. For example, a subscription to `Sensor/#` will receive messages published to `Sensor/`, `Sensor/temp`, `Sensor/temp/room1`, but not the messages published to `Sensor`.
+	Matches exactly one item in the topic hierarchy. For example, a subscription to `Sensor/+/room1` will receive messages published to `Sensor/temp/room1`, `Sensor/moisture/room1`, and so on.

Reserved Topics

Any topics beginning with $ are considered reserved and are not supported for publishing and subscribing except for those topics listed below. Any other attempts to publish or subscribe on topics beginning with $ will result in a terminated connection.

Topic	Allowed Operations	Description
$aws/events/presence/connected/*clientId*	Subscribe	AWS IoT publishes to this topic when an MQTT client with the specified client ID connects to AWS IoT. For more information, see Connect/Disconnect Events (p. 160).
$aws/events/presence/disconnected/*clientId*	Subscribe	AWS IoT publishes to this topic when an MQTT client with the specified client ID disconnects to AWS IoT. For more information, see Connect/Disconnect Events (p. 160).
$aws/events/subscriptions/subscribed/*clientId*	Subscribe	AWS IoT publishes to this topic when an MQTT client with the specified client ID subscribes to an MQTT topic. For more information, see Subscribe/Unsubscribe Events (p. 160).
$aws/events/subscriptions/unsubscribed/*clientId*	Subscribe	AWS IoT publishes to this topic when an MQTT client with the specified client ID unsubscribes to an MQTT topic. For more information, see Subscribe/Unsubscribe Events (p. 160).
$aws/things/*thingName*/shadow/delete	Publish/Subscribe	A thing or an application publishes to this topic to delete a thing shadow. For more information see http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#delete-pub-sub-topic.
$aws/things/*thingName*/shadow/delete/accepted	Subscribe	The Thing Shadows service sends messages to this topic when a thing shadow is deleted. For more information, see http://docs.aws.amazon.com/iot/latest/developerguide//

Topic	Allowed Operations	Description
		thing-shadow-mqtt.html#delete-accepted-pub-sub-topic.
$aws/things/*thingName*/shadow/delete/rejected	Subscribe	The Thing Shadows service sends messages to this topic when a request to delete a thing shadow is rejected. For more information, see http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#delete-rejected-pub-sub-topic.
$aws/things/*thingName*/shadow/get	Publish/Subscribe	An application or a thing publishes an empty message to this topic to get a thing shadow. For more information, see http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html.
$aws/things/*thingName*/shadow/get/accepted	Subscribe	The Thing Shadows service sends messages to this topic when a request for a thing shadow is made successfully. For more information, see http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#get-accepted-pub-sub-topic.
$aws/things/*thingName*/shadow/get/rejected	Subscribe	The Thing Shadows service sends messages to this topic when a request for a thing shadow is rejected. For more information, see http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#get-rejected-pub-sub-topic.
$aws/things/*thingName*/shadow/update	Publish/Subscribe	A thing or application publishes to this topic to update a thing shadow. For more information, see http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#update-pub-sub-topic.

Topic	Allowed Operations	Description
$aws/things/*thingName*/shadow/update/accepted	Subscribe	The Thing Shadows service sends messages to this topic when an update is successfully made to a thing shadow. For more information, see http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#update-accepted-pub-sub-topic.
$aws/things/*thingName*/shadow/update/rejected	Subscribe	The Thing Shadows service sends messages to this topic when an update to a thing shadow is rejected. For more information, see http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#update-rejected-pub-sub-topic.
$aws/things/*thingName*/shadow/update/delta	Subscribe	The Thing Shadows service sends messages to this topic when a difference is detected between the reported and desired sections of a thing shadow. For more information, see http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#update-delta-pub-sub-topic.
$aws/things/*thingName*/shadow/update/documents	Subscribe	AWS IoT publishes a state document to this topic whenever an update to the shadow is successfully performed. For more information, see http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#update-documents-pub-sub-topic.

Lifecycle Events

AWS IoT publishes lifecycle events on the MQTT topics discussed in the following sections. These messages allow you to be notified of lifecycle events from the message broker.

Note
Lifecycle messages might be sent out of order. You might receive duplicate messages.

Connect/Disconnect Events

AWS IoT publishes a message to the following MQTT topics when a client connects or disconnects:

```
$aws/events/presence/connected/clientId
```

or

```
$aws/events/presence/disconnected/clientId
```

Where `clientId` is the MQTT client ID that connects to or disconnects from the AWS IoT message broker.

The message published to this topic has the following structure:

```
{
    "clientId": "a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6",
    "timestamp": 1460065214626,
    "eventType": "connected",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "000000000000/ABCDEFGHIJKLMNOPQRSTU:some-user/
ABCDEFGHIJKLMNOPQRSTU:some-user"
}
```

The following is a list of JSON elements that are contained in the connection/disconnection messages published to the `$aws/events/presence/connected/clientId` topic.

clientId

The client ID of the connecting or disconnecting client.

Note
Client IDs that contain # or + do not receive lifecycle events.

eventType

The type of event. Valid values are `connected` or `disconnected`.

principalIdentifier

The credential used to authenticate. For TLS mutual authentication certificates, this is the certificate ID. For other connections, this is IAM credentials.

sessionIdentifier

A globally unique identifier in AWS IoT that exists for the life of the session.

timestamp

An approximation of when the event occurred, expressed in milliseconds since the Unix epoch. The accuracy of the timestamp is +/- 2 minutes.

Subscribe/Unsubscribe Events

AWS IoT publishes a message to the following MQTT topic when a client subscribes or unsubscribes to an MQTT topic:

```
$aws/events/subscriptions/subscribed/clientId
```

or

```
$aws/events/subscriptions/unsubscribed/clientId
```

Where `clientId` is the MQTT client ID that connects to the AWS IoT message broker.

The message published to this topic has the following structure:

```
{
    "clientId": "186b5",
    "timestamp": 1460065214626,
    "eventType": "subscribed" | "unsubscribed",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "000000000000/ABCDEFGHIJKLMNOPQRSTU:some-user/
ABCDEFGHIJKLMNOPQRSTU:some-user"
    "topics" : ["foo/bar","device/data","dog/cat"]
}
```

The following is a list of JSON elements that are contained in the subscribed and unsubscribed messages published to the `$aws/events/subscriptions/subscribed/clientId` and `$aws/events/subscriptions/unsubscribed/clientId` topics.

clientId

The client ID of the subscribing or unsubscribing client.

Note
Client IDs that contain # or + do not receive lifecycle events.

eventType

The type of event. Valid values are `subscribed` or `unsubscribed`.

principalIdentifier

The credential used to authenticate. For TLS mutual authentication certificates, this is the certificate ID. For other connections, this is IAM credentials.

sessionIdentifier

A globally unique identifier in AWS IoT that exists for the life of the session.

timestamp

An approximation of when the event occurred, expressed in milliseconds since the Unix epoch. The accuracy of the timestamp is +/- 2 minutes.

topics

An array of the MQTT topics to which the client has subscribed.

Note
Lifecycle messages might be sent out of order. You might receive duplicate messages.

Rules for AWS IoT

Rules give your devices the ability to interact with AWS services. Rules are analyzed and actions are performed based on the MQTT topic stream. You can use rules to support tasks like these:

- Augment or filter data received from a device.
- Write data received from a device to an Amazon DynamoDB database.
- Save a file to Amazon S3.
- Send a push notification to all users using Amazon SNS.
- Publish data to an Amazon SQS queue.
- Invoke a Lambda function to extract data.
- Process messages from a large number of devices using Amazon Kinesis.
- Send data to the Amazon Elasticsearch Service.
- Capture a CloudWatch metric.
- Change a CloudWatch alarm.
- Send the data from an MQTT message to Amazon Machine Learning to make predictions based on an Amazon ML model.
- Send a message to a Salesforce IoT Input Stream.

Before AWS IoT can perform these actions, you must grant it permission to access your AWS resources on your behalf. When the actions are performed, you incur the standard charges for the AWS services you use.

Contents

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 access to the required AWS resources. AWS IoT assumes this role when executing a rule.

To create an IAM role (AWS CLI)

1. Save the following trust policy document, which grants AWS IoT permission to assume the role, to a file called iot-role-trust.json:

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect": "Allow",
        "Principal": {
            "Service": "iot.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }]
}
```

Use the create-role command to create an IAM role specifying the iot-role-trust.json file:

```
aws iam create-role --role-name my-iot-role --assume-role-policy-document file://iot-role-trust.json
```

The output of this command looks like the following:

```
{
  "Role": {
      "AssumeRolePolicyDocument": "url-encoded-json",
      "RoleId": "AKIAIOSFODNN7EXAMPLE",
      "CreateDate": "2015-09-30T18:43:32.821Z",
      "RoleName": "my-iot-role",
      "Path": "/",
      "Arn": "arn:aws:iam::123456789012:role/my-iot-role"
  }
}
```

2. Save the following JSON into a file named iot-policy.json.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "dynamodb:*",
        "Resource": "*"
    }]
}
```

This JSON is an example policy document that grants AWS IoT administrator access to DynamoDB.

Use the create-policy command to grant AWS IoT access to your AWS resources upon assuming the role, passing in the iot-policy.json file:

```
aws iam create-policy --policy-name my-iot-policy --policy-document file://my-iot-policy-document.json
```

For more information about how to grant access to AWS services in policies for AWS IoT, see Creating an AWS IoT Rule (p. 164).

The output of the create-policy command contains the ARN of the policy. You need to attach the policy to a role.

```
{
    "Policy": {
        "PolicyName": "my-iot-policy",
        "CreateDate": "2015-09-30T19:31:18.620Z",
```

```
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "ZXR6A36LTYANPAI7NJ5UV",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:policy/my-iot-policy",
        "UpdateDate": "2015-09-30T19:31:18.620Z"
    }
}
```

3. Use the attach-role-policy command to attach your policy to your role:

```
aws iam attach-role-policy --role-name my-iot-role --policy-arn
 "arn:aws:iam::123456789012:policy/my-iot-policy"
```

Pass Role Permissions

Part of a rule definition is an IAM role that grants permission to access resources specified in the rule's action. The rules engine assumes that role when the rule's action is triggered. The role must be defined in the same AWS account as the rule.

When creating or replacing a rule you are, in effect, passing a role to the rules engine. The user performing this operation requires the `iam:PassRole` permission. To ensure you have this permission, create a policy that grants the `iam:PassRole` permission and attach it to your IAM user. The following policy shows how to allow `iam:PassRole` permission for a role.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

In this policy example, the `iam:PassRole` permission is granted for the role `myRole`. The role is specified using the role's ARN. You must attach this policy to your IAM user or role to which your user belongs. For more information, see Working with Managed Policies.

> **Note**
> Lambda functions use resource-based policy, where the policy is attached directly to the Lambda function itself. When creating a rule that invokes a Lambda function, you do not pass a role, so the user creating the rule does not need the `iam:PassRole` permission. For more information about Lambda function authorization, see Granting Permissions Using a Resource Policy.

Creating an AWS IoT Rule

You configure rules to route data from your connected things. Rules consist of the following:

Rule name

The name of the rule.

Optional description

A textual description of the rule.

SQL statement

A simplified SQL syntax to filter messages received on an MQTT topic and push the data elsewhere. For more information, see AWS IoT SQL Reference (p. 182).

SQL version

The version of the SQL rules engine to use when evaluating the rule. Although this property is optional, we strongly recommend that you specify the SQL version. If this property is not set, the default, `2015-10-08`, is used.

One or more actions

The actions AWS IoT performs when executing the rule. For example, you can insert data into a DynamoDB table, write data to an Amazon S3 bucket, publish to an Amazon SNS topic, or invoke a Lambda function.

An error action

The action AWS IoT performs when it is unable to perform a rule's action.

When you create a rule, be aware of how much data you are publishing on topics. If you create rules that include a wildcard topic pattern, they might match a large percentage of your messages, and you might need to increase the capacity of the AWS resources used by the target actions. Also, if you create a republish rule that includes a wildcard topic pattern, you can end up with a circular rule that causes an infinite loop.

Note
Creating and updating rules are administrator-level actions. Any user who has permission to create or update rules is able to access data processed by the rules.

To create a rule (AWS CLI)

Use the create-topic-rule command to create a rule:

```
aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://my-rule.json
```

The following is an example payload file with a rule that inserts all messages sent to the `iot/test` topic into the specified DynamoDB table. The SQL statement filters the messages and the role ARN grants AWS IoT permission to write to the DynamoDB table.

```
{
  "sql": "SELECT * FROM 'iot/test'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [{
      "dynamoDB": {
          "tableName": "my-dynamodb-table",
          "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
          "hashKeyField": "topic",
          "hashKeyValue": "${topic(2)}",
          "rangeKeyField": "timestamp",
          "rangeKeyValue": "${timestamp()}"
      }
```

```
  }]
}
```

The following is an example payload file with a rule that inserts all messages sent to the `iot/test` topic into the specified S3 bucket. The SQL statement filters the messages, and the role ARN grants AWS IoT permission to write to the Amazon S3 bucket.

```
{
    "awsIotSqlVersion": "2016-03-23",
    "sql": "SELECT * FROM 'iot/test'",
    "ruleDisabled": false,
    "actions": [
    {
        "s3": {
            "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",
            "bucketName": "my-bucket",
            "key": "myS3Key"
        }
    }
    ]
}
```

The following is an example payload file with a rule that pushes data to Amazon ES:

```
{
   "sql":"SELECT *, timestamp() as timestamp FROM 'iot/test'",
   "ruleDisabled":false,
  "awsIotSqlVersion": "2016-03-23",
    "actions":[
      {
        "elasticsearch":{
          "roleArn":"arn:aws:iam::123456789012:role/aws_iot_es",
          "endpoint":"https://my-endpoint",
         "index":"my-index",
          "type":"my-type",
          "id":"${newuuid()}"
        }
      }
    ]
}
```

The following is an example payload file with a rule that invokes a Lambda function:

```
{
  "sql": "expression",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [{
      "lambda": {
          "functionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-lambda-
function"
      }
  }]
}
```

The following is an example payload file with a rule that publishes to an Amazon SNS topic:

```
{
  "sql": "expression",
  "ruleDisabled": false,
```

```
  "awsIotSqlVersion": "2016-03-23",
  "actions": [{
      "sns": {
          "targetArn": "arn:aws:sns:us-west-2:123456789012:my-sns-topic",
          "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
      }
  }]
}
```

The following is an example payload file with a rule that republishes on a different MQTT topic:

```
{
  "sql": "expression",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [{
      "republish": {
          "topic": "my-mqtt-topic",
          "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
      }
  }]
}
```

The following is an example payload file with a rule that pushes data to an Amazon Kinesis Data Firehose stream:

```
{
    "sql": "SELECT * FROM 'my-topic'",
    "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
    "actions": [{
        "firehose": {
            "roleArn": ""arn:aws:iam::123456789012:role/my-iot-role",
            "deliveryStreamName": "my-stream-name"
        }
    }]
}
```

The following is an example payload file with a rule that uses the Amazon Machine Learning `machinelearning_predict` function to republish to a topic if the data in the MQTT payload is classified as a 1.

```
{
    "sql": "SELECT * FROM 'iot/test' where machinelearning_predict('my-model',
 'arn:aws:iam::123456789012:role/my-iot-aml-role', *).predictedLabel=1",
    "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
    "actions": [{
        "republish": {
            "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
            "topic": "my-mqtt-topic"
        }
    }]
}
```

The following is an example payload file with a rule that publishes messages to a Salesforce IoT Cloud input stream.

```
{
    "sql": "expression",
```

```
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [{
        "salesforce": {
            "token": "ABCDEFGHI123456789abcdefghi123456789",
            "url": "https://ingestion-cluster-id.my-env.sfdcnow.com/streams/stream-id/
connection-id/my-event"
        }
    }]
}
```

Viewing Your Rules

Use the list-topic-rules command to list your rules:

```
aws iot list-topic-rules
```

Use the get-topic-rule command to get information about a rule:

```
aws iot get-topic-rule --rule-name my-rule
```

SQL Versions

The AWS IoT rules engine uses an SQL-like syntax to select data from MQTT messages. The SQL statements are interpreted based on an SQL version specified with the `awsIotSqlVersion` property in a JSON document that describes the rule. For more information about the structure of JSON rule documents, see Creating a Rule (p. 164). The `awsIotSqlVersion` property allows you to specify which version of the AWS IoT SQL rules engine you want to use. When a new version is deployed, you can continue to use an older version or change your rule to use the new version. Your current rules continue to use the version with which they were created.

The following JSON example shows how to specify the SQL version using the `awsIotSqlVersion` property:

```
{
    "sql": "expression",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [{
        "republish": {
            "topic": "my-mqtt-topic",
            "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
        }
     }]
}
```

Current supported versions are:

- `2015-10-08`, the original SQL version built on 2015-10-08.
- `2016-03-23`, the SQL version built on 2016-03-23.
- `beta`, the most recent beta SQL version. The use of this version might introduce breaking changes to your rules.

What's New in the 2016-03-23 SQL Rules Engine Version

- Fixes for selecting nested JSON objects.
- Fixes for array queries.
- Inter-object query support.
- Support to output an array as a top-level object.
- Adds the encode (*value*, *encodingScheme*) function, which can be applied on both JSON and non-JSON format data.

Inter-Object Queries

This feature allows you to query for an attribute in a JSON object. For example, given the following MQTT message:

```
{
    "e": [
       { "n": "temperature", "u": "Cel", "t": 1234, "v":22.5 },
       { "n": "light", "u": "lm", "t": 1235, "v":135 },
       { "n": "acidity", "u": "pH", "t": 1235, "v":7 }
    ]
}
```

And the following rule:

```
SELECT (SELECT v FROM e WHERE n = 'temperature') as temperature FROM 'my/topic'
```

The rule generates the following output:

```
{"temperature": [{"v":22.5}]}
```

Using the same MQTT message, given a slightly more complicated rule such as:

```
SELECT get((SELECT v FROM e WHERE n = 'temperature'),1).v as temperature FROM 'topic'
```

The rule generates the following output:

```
{"temperature":22.5}
```

Output an `Array` as a Top-Level Object

This feature allows a rule to return an array as a top-level object. For example, given the following MQTT message:

```
{
    "a": {"b":"c"},
    "arr":[1,2,3,4]
}
```

And the following rule:

```
SELECT VALUE arr FROM 'topic'
```

The rule generates the following output:

```
[1,2,3,4]
```

Encode Function

Encodes the payload, which potentially might be non-JSON data, into its string representation based on the specified encoding scheme.

Troubleshooting a Rule

If you are having an issue with your rules, you should enable CloudWatch Logs. By analyzing your logs, you can determine whether the issue is authorization or whether, for example, a WHERE clause condition did not match. For more information about using Amazon CloudWatch Logs, see Setting Up CloudWatchLogs.

Rule Error Handling

When AWS IoT receives a message from a device, the Rules Engine checks to see if the message matches a rule. If so, the rule's SQL statement is evaluated and the rule's actions are invoked, passing the SQL statement's result.

If a problem occurs when invoking an action, the Rules Engine will invoke an error action, if one is specified for the rule. This may happen when, for example:

- A rule doesn't have permission to access an Amazon S3 bucket.
- A user error causes DynamoDB provisioned throughput to be exceeded.

Error Action Message Format

A single message is generated per rule and message. For example, if two rule actions in the same rule fail, the error action will receive one message containing both errors.

The error action message will look like this:

```
{
      "ruleName": "TestAction",
      "topic": "testme/action",
      "cloudwatchTraceId": "7e146a2c-95b5-6caf-98b9-50e3969734c7",
      "clientId": "iotconsole-1511213971966-0",
      "base64OriginalPayload":
 "ewogICJtZXNzYWdlIjogIkhlbGxvIHZyb20gQVdTIElvVCBjb25zb2xlIgp9",
      "failures": [
        {
          "failedAction": "S3Action",
          "failedResource": "us-east-1-s3-verify-user",
          "errorMessage": "Failed to put S3 object. The error received was The
 specified bucket does not exist (Service: Amazon S3; Status Code: 404; Error
 Code: NoSuchBucket; Request ID: 9DF5416B9B47B9AF; S3 Extended Request ID:
 yMah1cwPhqTH267QLPhTKeVPKJB8BO5ndBHzOmWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y=).
```

```
 Message arrived on: error/action, Action: s3, Bucket: us-
east-1-s3-verify-user, Key: \"aaa\". Value of x-amz-id-2:
 yMah1cwPhqTH267QLPhTKeVPKJB8BO5ndBHzOmWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y="
        }
      ]
    }
```

ruleName

The name of the rule that triggered the error action.

topic

The topic on which the original message was received.

cloudwatchTraceId

A unique identity referring to the error logs in CloudWatch.

clientId

The client ID of the message publisher.

base64OriginalPayload

The original message payload base64 encoded.

failures

failedAction

The name of the action that failed to complete, for example "S3Action".

failedResource

The name of the resource, for example the name of an S3 bucket.

errorMessage

The description and explanation of the error.

Error Action Example

Here is an example of a rule with an added error action. The following rule has an action that writes message data to a DynamoDB table and an error action that writes data to an Amazon S3 bucket:

```
{
    "sql" : "SELECT * FROM ..."
    "actions" : [{
        "dynamoDB" : {
            "table" : "PoorlyConfiguredTable",
            "hashKeyField" : "AConstantString",
            "hashKeyValue" : "AHashKey"}}
    ],
    "errorAction" : { "s3" : {
            "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",
            "bucket" : "message-processing-errors",
            "key" : "${replace(topic(), '/', '-') + '-' + timestamp() + '-' + newuuid()}"
    }}
}
```

You can use any function or substitution in an error action's SQL statement, except for external functions (for example, `get_thing_shadow`, `aws_lambda`, and `machinelearning_predict`.)

For more information about rules and how to specify an error action, see Creating an AWS IoT Rule.

For more information on using CloudWatch to monitor the success or failure of rules, see AWS IoT Metrics and Dimensions (p. 333).

Deleting a Rule

When you are finished with a rule, you can delete it.

To delete a rule (AWS CLI)

Use the delete-topic-rule command to delete a rule:

```
aws iot delete-topic-rule --rule-name my-rule
```

AWS IoT Rule Actions

AWS IoT rule actions are used to specify what to do when a rule is triggered. You can define actions to write data to a DynamoDB database or a Kinesis stream or to invoke a Lambda function, and more. The following actions are supported:

- `cloudwatchAlarm` to change a CloudWatch alarm.
- `cloudwatchMetric` to capture a CloudWatch metric.
- `dynamoDB` to write data to a DynamoDB database.
- `dynamoDBv2` to write data to a DynamoDB database.
- `elasticsearch` to write data to an Amazon Elasticsearch Service domain.
- `firehose` to write data to an Amazon Kinesis Data Firehose stream.
- `kinesis` to write data to a Kinesis stream.
- `lambda` to invoke a Lambda function.
- `s3` to write data to an Amazon S3 bucket.
- `sns` to write data as a push notification.
- `sqs` to write data to an SQS queue.
- `republish` to republish the message on another MQTT topic.
- `salesforce` to write a message to a Salesforce IoT input stream.

> **Note**
> The AWS IoT rules engine does not currently retry delivery for messages that fail to be published to another service.

The following sections discuss each action in detail.

CloudWatch Alarm Action

The CloudWatch alarm action allows you to change CloudWatch alarm state. You can specify the state change reason and value in this call. When creating an AWS IoT rule with a CloudWatch alarm action, you must specify the following information:

roleArn

The IAM role that allows access to the CloudWatch alarm.

alarmName

The CloudWatch alarm name.

stateReason

Reason for the alarm change.

stateValue

The value of the alarm state. Acceptable values are `OK`, `ALARM`, `INSUFFICIENT_DATA`.

Note
Ensure the role associated with the rule has a policy that grants the `cloudwatch:SetAlarmState` permission.

The following JSON example shows how to define a CloudWatch alarm action in an AWS IoT rule:

```
{
    "rule": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "actions": [{
            "cloudwatchAlarm": {
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw",
                "alarmName": "IotAlarm",
                "stateReason": "Temperature stabilized.",
                "stateValue": "OK"
            }
        }]
    }
}
```

For more information, see CloudWatch Alarms.

CloudWatch Metric Action

The CloudWatch metric action allows you to capture a CloudWatch metric. You can specify the metric namespace, name, value, unit, and timestamp. When creating an AWS IoT rule with a CloudWatch metric action, you must specify the following information:

roleArn

The IAM role that allows access to the CloudWatch metric.

metricNamespace

CloudWatch metric namespace name.

metricName

The CloudWatch metric name.

metricValue

The CloudWatch metric value.

metricUnit

The metric unit supported by CloudWatch.

metricTimestamp

An optional Unix timestamp.

Note
Ensure the role associated with the rule has a policy granting the `cloudwatch:PutMetricData` permission.

The following JSON example shows how to define a CloudWatch metric action in an AWS IoT rule:

```
{
    "rule": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "actions": [{
            "cloudwatchMetric": {
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw",
                "metricNamespace": "IotNamespace",
                "metricName": "IotMetric",
                "metricValue": "1",
                "metricUnit": "Count",
                "metricTimestamp": "1456821314"
            }
        }]
    }
}
```

For more information, see CloudWatch Metrics.

DynamoDB Action

The `dynamoDB` action allows you to write all or part of an MQTT message to a DynamoDB table. When creating a DynamoDB rule, you must specify the following information:

hashKeyType

The data type of the hash key (also called the partition key). Valid values are: `"STRING"` or `"NUMBER"`.

hashKeyField

The name of the hash key (also called the partition key).

hashKeyValue

The value of the hash key.

rangeKeyType

Optional. The data type of the range key (also called the sort key). Valid values are: `"STRING"` or `"NUMBER"`.

rangeKeyField

Optional. The name of the range key (also called the sort key).

rangeKeyValue

Optional. The value of the range key.

operation

Optional. The type of operation to be performed. This follows the substitution template, so it can be `${operation}`, but the substitution must result in one of the following: `INSERT`, `UPDATE`, or `DELETE`.

payloadField

Optional. The name of the field where the payload is written. If this value is omitted, the payload is written to the `payload` field.

table

The name of the DynamoDB table.

roleARN

The IAM role that allows access to the DynamoDB table. At a minimum, the role must allow the `dynamoDB:PutItem` IAM action.

The data written to the DynamoDB table is the result from the SQL statement of the rule. The `hashKeyValue` and `rangeKeyValue` fields are usually composed of expressions (for example, "${topic()}" or "${timestamp()}").

Note
Non-JSON data is written to DynamoDB as binary data. The DynamoDB console displays the data as Base64-encoded text.
Ensure the role associated with the rule has a policy granting the `dynamodb:PutItem` permission.

The following JSON example shows how to define a `dynamoDB` action in an AWS IoT rule:

```
{
    "rule": {
        "ruleDisabled": false,
        "sql": "SELECT * AS message FROM 'some/topic'",
        "description": "A test Dynamo DB rule",
        "actions": [{
            "dynamoDB": {
                "hashKeyField": "key",
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDB",
                 "tableName": "my_ddb_table",
                 "hashKeyValue": "${topic()}",
                 "rangeKeyValue": "${timestamp()}",
                 "rangeKeyField": "timestamp"
             }
        }]
    }
}
```

For more information, see the Amazon DynamoDB Getting Started Guide.

DynamoDBv2 Action

The `dynamoDBv2` action allows you to write all or part of an MQTT message to a DynamoDB table. Each attribute in the payload is written to a separate column in the DynamoDB database. When creating a DynamoDB rule, you must specify the following information:

roleARN

The IAM role that allows access to the DynamoDB table. At a minimum, the role must allow the `dynamoDB:PutItem` IAM action.

tableName

The name of the DynamoDB table.

Note
The MQTT message payload must contain a root-level key that matches the table's primary partition key and a root-level key that matches the table's primary sort key, if one is defined.

The data written to the DynamoDB table is the result from the SQL statement of the rule.

Note
Ensure the role associated with the rule has a policy granting the `dynamodb:PutItem` permission.

The following JSON example shows how to define a `dynamoDB` action in an AWS IoT rule:

```
{
    "rule": {
        "ruleDisabled": false,
        "sql": "SELECT * AS message FROM 'some/topic'",
        "description": "A test DynamoDBv2 rule",
        "actions": [{
            "dynamoDBv2": {
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDBv2",
                "putItem": {
                    "tableName": "my_ddb_table"
                }
             }
        }]
    }
}
```

For more information, see the Amazon DynamoDB Getting Started Guide.

Amazon ES Action

The `elasticsearch` action allows you to write data from MQTT messages to an Amazon Elasticsearch Service domain. Data in Amazon ES can then be queried and visualized by using tools like Kibana. When you create an AWS IoT rule with an `elasticsearch` action, you must specify the following information:

endpoint

The endpoint of your Amazon ES domain.

index

The Amazon ES index where you want to store your data.

type

The type of document you are storing.

id

The unique identifier for each document.

Note
Ensure the role associated with the rule has a policy granting the `es:ESHttpPut` permission.

The following JSON example shows how to define an `elasticsearch` action in an AWS IoT rule:

```
{
  "rule":{
   "sql":"SELECT *, timestamp() as timestamp FROM 'iot/test'",
   "ruleDisabled":false,
    "actions":[
      {
        "elasticsearch":{
          "roleArn":"arn:aws:iam::123456789012:role/aws_iot_es",
```

```
          "endpoint":"https://my-endpoint",
         "index":"my-index",
          "type":"my-type",
          "id":"${newuuid()}"
        }
      }
    ]
  }
}
```

For more information, see the Amazon ES Developer Guide.

Firehose Action

A `firehose` action sends data from an MQTT message that triggered the rule to a Kinesis Data Firehose stream. When creating a rule with a `firehose` action, you must specify the following information:

deliveryStreamName

The Kinesis Data Firehose stream to which to write the message data.

roleArn

The IAM role that allows access to Kinesis Data Firehose.

separator

A character separator that is used to separate records written to the Firehose stream. Valid values are: '\n' (newline), '\t' (tab), '\r\n' (Windows newline), ',' (comma).

Note
Make sure the role associated with the rule has a policy that grants the `firehose:PutRecord` permission.

The following JSON example shows how to create an AWS IoT rule with a `firehose` action:

```
{
    "rule": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "actions": [{
            "firehose": {
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_firehose",
                "deliveryStreamName": "my_firehose_stream"
            }
        }]
    }
}
```

For more information, see the Kinesis Data Firehose Developer Guide.

Kinesis Action

The `kinesis` action allows you to write data from MQTT messages into a Kinesis stream. When creating an AWS IoT rule with a `kinesis` action, you must specify the following information:

stream

The Kinesis stream to which to write data.

partitionKey

The partition key used to determine to which shard the data is written. The partition key is usually composed of an expression (for example, "${topic()}" or "${timestamp()}").

Note
Ensure that the policy associated with the rule has the `kinesis:PutRecord` permission.

The following JSON example shows how to define a `kinesis` action in an AWS IoT rule:

```
{
    "rule": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "actions": [{
            "kinesis": {
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_kinesis",
                "streamName": "my_kinesis_stream",
                "partitionKey": "${topic()}"
            }
        }],
    }
}
```

For more information, see the Kinesis Developer Guide.

Lambda Action

A `lambda` action calls a Lambda function, passing in the MQTT message that triggered the rule. In order for AWS IoT to call a Lambda function, you must configure a policy granting the `lambda:InvokeFunction` permission to AWS IoT. Lambda functions use resource-based policies, so you must attach the policy to the Lambda function itself. Use the following CLI command to attach a policy granting `lambda:InvokeFunction` permission:

```
aws lambda add-permission --function-name "function_name" --region "region" --principal
 iot.amazonaws.com --source-arn arn:aws:iot:us-east-2:account_id:rule/rule_name --source-
account "account_id" --statement-id "unique_id" --action "lambda:InvokeFunction"
```

The following are the arguments for the `add-permission` command:

--function-name

Name of the Lambda function whose resource policy you are updating by adding a new permission.

--region

The AWS region of your account.

--principal

The principal who is getting the permission. This should be `iot.amazonaws.com` to allow AWS IoT permission to call a Lambda function.

--source-arn

The ARN of the rule. You can use the `get-topic-rule` CLI command to get the ARN of a rule.

--source-account

The AWS account where the rule is defined.

--statement-id

A unique statement identifier.

--action

The Lambda action you want to allow in this statement. In this case, we want to allow AWS IoT to invoke a Lambda function, so we specify `lambda:InvokeFunction`.

Note
If you add a permission for an AWS IoT principal without providing the source ARN, any AWS account that creates a rule with your Lambda action can trigger rules to invoke your Lambda function from AWS IoT

For more information, see Lambda Permission Model.

When creating a rule with a `lambda` action, you must specify the Lambda function to invoke when the rule is triggered.

The following JSON example shows a rule that calls a Lambda function:

```
{
    "rule": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "actions": [{
            "lambda": {
                "functionArn": "arn:aws:lambda:us-
east-2:123456789012:function:myLambdaFunction"
             }
        }]
    }
}
```

For more information, see the AWS Lambda Developer Guide.

Republish Action

The `republish` action allows you to republish the message that triggered the role to another MQTT topic. When creating a rule with a `republish` action, you must specify the following information:

topic

The MQTT topic to which to republish the message.

roleArn

The IAM role that allows publishing to the MQTT topic.

Note
Make sure the role associated with the rule has a policy granting the `iot:Publish` permission.

```
{
    "rule": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "actions": [{
            "republish": {
                "topic": "another/topic",
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish"
```

```
            }
        }]
    }
}
```

S3 Action

A `s3` action writes the data from the MQTT message that triggered the rule to an Amazon S3 bucket. When creating an AWS IoT rule with an `s3` action, you must specify the following information:

bucket

The Amazon S3 bucket to which to write data.

cannedacl

The Amazon S3 canned ACL that controls access to the object identified by the object key. For more information, see S3 Canned ACLs.

key

The path to the file where the data is written. For example, if the value of this argument is "${topic()}/${timestamp()}", the topic the message was sent to is "this/is/my/topic,", and the current timestamp is 1460685389, the data is written to a file called "1460685389" in the "this/is/my/topic" folder on Amazon S3.

Note
Using a static key results in a single file in Amazon S3 being overwritten for each invocation of the rule. More common use cases are to use the message timestamp or another unique message identifier, so that a new file is saved in Amazon S3 for each message received.

roleArn

The IAM role that allows access to the Amazon S3 bucket.

Note
Make sure the role associated with the rule has a policy granting the `s3:PutObject` permission.

The following JSON example shows how to define an `s3` action in an AWS IoT rule:

```
{
    "rule": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "actions": [{
            "s3": {
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",
                "bucketName": "my-bucket",
                "key": "${topic()}/${timestamp()}"
            }
        }]
    }
}
```

For more information, see the Amazon S3 Developer Guide.

SNS Action

A `sns` action sends the data from the MQTT message that triggered the rule as an SNS push notification. When creating a rule with an `sns` action, you must specify the following information:

messageFormat

The message format. Accepted values are "JSON" and "RAW." The default value of the attribute is "RAW." SNS uses this setting to determine if the payload should be parsed and relevant platform-specific parts of the payload should be extracted.

roleArn

The IAM role that allows access to SNS.

targetArn

The SNS topic or individual device to which the push notification is sent.

Note
Make sure the policy associated with the rule has the `sns:Publish` permission.

The following JSON example shows how to define an `sns` action in an AWS IoT rule:

```
{
    "rule": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "actions": [{
            "sns": {
                "targetArn": "arn:aws:sns:us-east-2:123456789012:my_sns_topic",
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sns"
            }
        }]
    }
}
```

For more information, see the Amazon SNS Developer Guide.

SQS Action

A `sqs` action sends data from the MQTT message that triggered the rule to an SQS queue. When creating a rule with an `sqs` action, you must specify the following information:

queueUrl

The URL of the SQS queue to which to write the data.

useBase64

Set to `true` if you want the MQTT message data to be Base64-encoded before writing to the SQS queue. Otherwise, set to `false`.

roleArn

The IAM role that allows access to the SQS queue.

Note
Make sure the role associated with the rule has a policy granting the `sqs:SendMessage` permission.

The following JSON example shows how to create an AWS IoT rule with an `sqs` action:

```
{
    "rule": {
```

```
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "actions": [{
            "sqs": {
                "queueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/
my_sqs_queue",
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sqs",
                "useBase64": false
            }
        }]
    }
}
```

For more information, see the Amazon SQS Developer Guide.

Salesforce Action

A `salesforce` action sends data from the MQTT message that triggered the rule to a Salesforce IoT Input Stream. When creating a rule with a `salesforce` action, you must specify the following information:

url

The URL exposed by the Salesforce IoT Input Stream. The URL is available from the Salesforce IoT Platform when you create an Input Stream. Refer to the Salesforce IoT documentation to learn more.

token

The token used to authenticate access to the specified Salesforce IoT Input Stream. The token is available from the Salesforce IoT Platform when you create an Input Stream. Refer to the Salesforce IoT documentation to learn more.

Note
These parameters do not support substitution.

The following JSON example shows how to create an AWS IoT rule with a `salesforce` action:

```
{
    "sql": "expression",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [{
        "salesforce": {
            "token": "ABCDEFGHI123456789abcdefghi123456789",
            "url": "https://ingestion-cluster-id.my-env.sfdcnow.com/streams/stream-id/
connection-id/my-event"
        }
    }]
}
```

For more information, refer to the Salesforce IoT documentation.

AWS IoT SQL Reference

In AWS IoT, rules are defined using an SQL-like syntax. SQL statements are composed of three types of clauses:

SELECT

Required. Extracts information from the incoming payload and performs transformations.

FROM

Required. The MQTT topic filter from which the rule receives messages.

WHERE

Optional. Adds conditional logic that determines if a rule is evaluated and its actions are executed.

An example SQL statement looks like this:

```
SELECT color AS rgb FROM 'a/b' WHERE temperature > 50
```

An example MQTT message (also called an incoming payload) looks like this:

```
{
    "color":"red",
    "temperature":100
}
```

If this message is published on the `'a/b'` topic, the rule is triggered and the SQL statement is evaluated. The SQL statement extracts the value of the `rgb` property if the `"temperature"` property is greater than 50. The WHERE clause specifies the condition `temperature > 50`. The `AS` keyword renames the `"color"` property to `"rgb"`. The result (also called an outgoing payload) looks like this:

```
{
    "rgb":"red"
}
```

This data is then forwarded to the rule's action, which sends the data for more processing. For more information about rule actions, see AWS IoT Rule Actions (p. 172).

Data Types

The AWS IoT rules engine supports all JSON data types.

Supported Data Types

Type	Meaning
`Int`	A discrete `Int`. 34 digits maximum.
`Decimal`	A `Decimal` with a precision of 34 digits, with a minimum non-zero magnitude of 1E-999 and a maximum magnitude 9.999…E999. **Note** Some functions return `Decimals` with double precision rather than 34-digit precision.
`Boolean`	`True` or `False`.
`String`	A UTF-8 string.
`Array`	A series of values that don't have to have the same type.

Type	Meaning
`Object`	A JSON value consisting of a key and a value. Keys must be strings. Values can be any type.
`Null`	`Null` as defined by JSON. It's an actual value that represents the absence of a value. You can explicitly create a `Null` value by using the `Null` keyword in your SQL statement. For example: `"SELECT NULL AS n FROM 'a/b'"`

`Undefined`	Not a value. This isn't explicitly representable in JSON except by omitting the value. For example, in the object `{"foo": null}`, the key "foo" returns NULL, but the key "bar" returns `Undefined`. Internally, the SQL language treats `Undefined` as a value, but it isn't representable in JSON, so when serialized to JSON, the results are `Undefined`.
	`{"foo":null, "bar":undefined}`
	is serialized to JSON as:
	`{"foo":null}`
	Similarly, `Undefined` is converted to an empty string when serialized by itself. Functions called with invalid arguments (for example, wrong types, wrong number of arguments, and so on) returns `Undefined`.

Conversions

The following table lists the results when a value of one type is converted to another type (when a value of the incorrect type is given to a function). For example, if the absolute value function "abs" (which expects an `Int` or `Decimal`) is given a `String`, it attempts to convert the `String` to a `Decimal`, following these rules. In this case, 'abs("-5.123")' is treated as 'abs(-5.123)'.

> **Note**
> There are no attempted conversions to `Array`, `Object`, `Null`, or `Undefined`.

To Decimal

Argument Type	Result
`Int`	A `Decimal` with no decimal point.
`Decimal`	The source value.
`Boolean`	`Undefined`. (You can explicitly use the cast function to transform true = 1.0, false = 0.0.)
`String`	The SQL engine tries to parse the string as a `Decimal`. AWS IoT attempts to parse strings matching the regular expression:`^-?\d+(\.\d`

Argument Type	Result
	`+)?((?i)E-?\d+)?$`. "0", "-1.2", "5E-12" are all examples of strings that would be automatically converted to `Decimals`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Null`.
Undefined	`Undefined`.

To Int

Argument Type	Result
`Int`	The source value.
`Decimal`	The source value rounded to the nearest `Int`.
`Boolean`	`Undefined`. (You can explicitly use the cast function to transform true = 1.0, false = 0.0.)
`String`	The SQL engine will try to parse the string as a `Decimal`. We will attempt to parse strings matching the regular expression:`^-?\d+(\.\d+)?((?i)E-?\d+)?$`. "0", "-1.2", "5E-12" are all examples of strings that would automatically be converted to `Decimals`. We will attempt to convert the `String` to a `Decimal`, and then truncate the decimal places of that `Decimal` to make an `Int`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Null`.
Undefined	`Undefined`.

To Boolean

Argument Type	Result
`Int`	`Undefined`. (You can explicitly use the `cast` function to transform 0 = False, any_nonzero_value = True.)
`Decimal`	`Undefined`. (You can explicitly use the cast function to transform 0 = False, any_nonzero_value = True.)
`Boolean`	The original value.
`String`	"true"=True and "false"=False (case-insensitive). Other string values will be `Undefined`.

Argument Type	Result
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

To String

Argument Type	Result
`Int`	A string representation of the `Int` in standard notation.
`Decimal`	A string representing the `Decimal` value, possibly in scientific notation.
`Boolean`	"true" or "false". All lowercase.
`String`	The original value.
Array	The `Array` serialized to JSON. The resultant string will be a comma-separated list, enclosed in square brackets. `String`s will be quoted. `Decimal`s, `Int`s, `Boolean`s and `Null` will not.
Object	The object serialized to JSON. The resultant string will be a comma-separated list of key-value pairs and will begin and end with curly braces. `String`s will be quoted. `Decimal`s, `Int`s, `Boolean`s and `Null` will not.
Null	`Undefined`.
Undefined	Undefined.

Operators

The following operators can be used in SELECT, FROM, and WHERE clauses.

AND operator

Returns a `Boolean` result. Performs a logical AND operation. Returns true if left and right operands are true. Otherwise, returns false. `Boolean` operands or case-insensitive "true" or "false" string operands are required.

Syntax: *`expression`* `AND` *`expression`*.

AND Operator

Left Operand	Right Operand	Output
`Boolean`	`Boolean`	`Boolean`. True if both operands are true. Otherwise, false.

Left Operand	Right Operand	Output
`String/Boolean`	`String/Boolean`	If all strings are "true" or "false" (case-insensitive), they are converted to `Boolean` and processed normally as *`boolean`* `AND` *`boolean`*.
Other Value	Other Value	`Undefined`.

OR operator

Returns a `Boolean` result. Performs a logical OR operation. Returns true if either the left or the right operands are true. Otherwise, returns false. `Boolean` operands or case-insensitive "true" or "false" string operands are required.

Syntax: *`expression`* `OR` *`expression`*.

OR Operator

Left Operand	Right Operand	Output
`Boolean`	`Boolean`	`Boolean`. True if either operand is true. Otherwise, false.
`String/Boolean`	`String/Boolean`	If all strings are "true" or "false" (case-insensitive), they are converted to `Booleans` and processed normally as *`boolean`* `OR` *`boolean`*.
Other Value	Other Value	`Undefined`.

NOT operator

Returns a `Boolean` result. Performs a logical NOT operation. Returns true if the operand is false. Otherwise, returns true. A boolean operand or case-insensitive "true" or "false" string operand is required.

Syntax: `NOT` *`expression`*.

NOT Operator

Operand	Output
`Boolean`	`Boolean`. True if operand is false. Otherwise, true.
`String`	If string is "true" or "false" (case-insensitive), it is converted to the corresponding boolean value, and the opposite value is returned.
Other Value	`Undefined`.

> operator

Returns a `Boolean` result. Returns true if the left operand is greater than the right operand. Both operands are converted to a `Decimal`, and then compared.

Syntax: *`expression`* `>` *`expression`*.

> Operator

Left Operand	Right Operand	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. True if the left operand is greater than the right operand. Otherwise, false.
`String/Int/Decimal`	`String/Int/Decimal`	If all strings can be converted to `Decimal`, then `Boolean`. Returns true if the left operand is greater than the right operand. Otherwise, false.
Other Value	`Undefined`.	`Undefined`.

>= operator

Returns a `Boolean` result. Returns true if the left operand is greater than or equal to the right operand. Both operands are converted to a `Decimal`, and then compared.

Syntax: *`expression`* `>=` *`expression`*.

>= Operator

Left Operand	Right Operand	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. True if the left operand is greater than or equal to the right operand. Otherwise, false.
`String/Int/Decimal`	`String/Int/Decimal`	If all strings can be converted to `Decimal`, then `Boolean`. Returns true if the left operand is greater than or equal to the right operand. Otherwise, false.
Other Value	`Undefined`.	`Undefined`.

< operator

Returns a `Boolean` result. Returns true if the left operand is less than the right operand. Both operands are converted to a `Decimal`, and then compared.

Syntax: *`expression`* `<` *`expression`*.

< Operator

Left Operand	Right Operand	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. True if the left operand is less than the right operand. Otherwise, false.
`String/Int/Decimal`	`String/Int/Decimal`	If all strings can be converted to `Decimal`, then `Boolean`. Returns true if the left operand is less than the right operand. Otherwise, false.
Other Value	`Undefined`	`Undefined`

<= operator

Returns a `Boolean` result. Returns true if the left operand is less than or equal to the right operand. Both operands are converted to a `Decimal`, and then compared.

Syntax: *expression* <= *expression*.

>= Operator

Left Operand	Right Operand	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. True if the left operand is less than or equal to the right operand. Otherwise, false.
`String/Int/Decimal`	`String/Int/Decimal`	If all strings can be converted to `Decimal`, then `Boolean`. Returns true if the left operand is less than or equal to the right operand. Otherwise, false.
Other Value	`Undefined`	`Undefined`

<> operator

Returns a `Boolean` result. Returns true if both left and right operands are not equal. Otherwise, returns false.

Syntax: *expression* <> *expression*.

<> Operator

Left Operand	Right Operand	Output
`Int`	`Int`	True if left operand is not equal to right operand. Otherwise, false.
`Decimal`	`Decimal`	True if left operand is not equal to right operand. Otherwise, false.`Int` is converted to `Decimal` before being compared.
`String`	`String`	True if left operand is not equal to right operand. Otherwise, false.
Array	Array	True if the items in each operand are not equal and not in the same order. Otherwise, false
Object	Object	True if the keys and values of each operand are not equal. Otherwise, false. The order of keys/values is unimportant.
Null	Null	False.
Any Value	`Undefined`	Undefined.
`Undefined`	Any Value	Undefined.
Mismatched Type	Mismatched Type	True.

= operator

Returns a `Boolean` result. Returns true if both left and right operands are equal. Otherwise, returns false.

Syntax: *expression* = *expression*.

= Operator

Left Operand	Right Operand	Output
`Int`	`Int`	True if left operand is equal to right operand. Otherwise, false.
`Decimal`	`Decimal`	True if left operand is equal to right operand. Otherwise, false.`Int` is converted to `Decimal` before being compared.
`String`	`String`	True if left operand is equal to right operand. Otherwise, false.
Array	Array	True if the items in each operand are equal and in the same order. Otherwise, false.
Object	Object	True if the keys and values of each operand are equal. Otherwise, false. The order of keys/values is unimportant.
Any Value	`Undefined`	`Undefined`.
`Undefined`	Any Value	`Undefined`.
Mismatched Type	Mismatched Type	False.

+ operator

The "+" is an overloaded operator. It can be used for string concatenation or addition.

Syntax: *`expression`* `+` *`expression`*.

+ Operator

Left Operand	Right Operand	Output
`String`	Any Value	Converts the right operand to a string and concatenates it to the end of the left operand.
Any Value	`String`	Converts the left operand to a string and concatenates the right operand to the end of the converted left operand.
`Int`	`Int`	`Int` value. Adds operands together.
`Int/Decimal`	`Int/Decimal`	`Decimal` value. Adds operands together.
Other Value	Other Value	`Undefined`.

- operator

Subtracts the right operand from the left operand.

Syntax: *`expression`* `-` *`expression`*.

- Operator

Left Operand	Right Operand	Output
`Int`	`Int`	`Int` value. Subtracts right operand from left operand.

Left Operand	Right Operand	Output
`Int/Decimal`	`Int/Decimal`	`Decimal` value. Subtracts right operand from left operand.
`String/Int/Decimal`	`String/Int/Decimal`	If all strings convert to `Decimals` correctly, a `Decimal` value is returned. Subtracts right operand from left operand. Otherwise, returns `Undefined`.
Other Value	Other value	`Undefined`.
Other Value	Other Value	`Undefined`.

* operator

Multiplies the left operand by the right operand.

Syntax: *`expression`* `*` *`expression`*.

*** Operator**

Left Operand	Right Operand	Output
`Int`	`Int`	`Int` value. Multiplies the left operand by the right operand.
`Int/Decimal`	`Int/Decimal`	`Decimal` value. Multiplies the left operand by the right operand.
`String/Int/Decimal`	`String/Int/Decimal`	If all strings convert to `Decimals` correctly, a `Decimal` value is returned. Multiplies the left operand by the right operand. Otherwise, returns `Undefined`.
Other Value	Other value	`Undefined`.

/ operator

Divides the left operand by the right operand.

Syntax: *`expression`* `/` *`expression`*.

/ Operator

Left Operand	Right Operand	Output
`Int`	`Int`	`Int` value. Divides the left operand by the right operand.
`Int/Decimal`	`Int/Decimal`	`Decimal` value. Divides the left operand by the right operand.
`String/Int/Decimal`	`String/Int/Decimal`	If all strings convert to `Decimals` correctly, a `Decimal` value is returned. Divides the left operand by the right operand. Otherwise, returns `Undefined`.
Other Value	Other value	`Undefined`.

% operator

Returns the remainder from dividing the left operand by the right operand.

Syntax: *expression* % *expression*.

% Operator

Left Operand	Right Operand	Output
Int	Int	Int value. Returns the remainder from dividing the left operand by the right operand.
String/Int/Decimal	String/Int/Decimal	If all Strings convert to Decimals correctly, a Decimal value is returned. Returns the remainder from dividing the left operand by the right operand. Otherwise, Undefined.
Other Value	Other value	Undefined.

Functions

You can use the following built-in functions in the SELECT or WHERE clauses of your SQL expressions.

abs(Decimal)

Returns the absolute value of a number. Supported by SQL version 2015-10-8 and later.

Example: `abs(-5)` returns 5.

Argument Type	Result
Int	Int, the absolute value of the argument.
Decimal	Decimal, the absolute value of the argument.
Boolean	Undefined.
String	Decimal. The result is the absolute value of the argument. If the string cannot be converted, the result is Undefined.
Array	Undefined.
Object	Undefined.
Null	Undefined.
Undefined	Undefined.

accountid()

Returns the ID of the account that owns this rule as a String. Supported by SQL version 2015-10-8 and later.

Example:

```
accountid()  = "123456789012"
```

acos(Decimal)

Returns the inverse cosine of a number in radians. `Decimal` arguments are rounded to double precision before function application. Supported by SQL version 2015-10-8 and later.

Example: `acos(0)` = 1.5707963267948966

Argument Type	Result
`Int`	`Decimal` (with double precision), the inverse cosine of the argument. Imaginary results are returned as `Undefined`.
`Decimal`	`Decimal` (with double precision), the inverse cosine of the argument. Imaginary results are returned as `Undefined`.
`Boolean`	`Undefined`.
`String`	`Decimal`, the inverse cosine of the argument. If the string cannot be converted, the result is `Undefined`. Imaginary results are returned as `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

asin(Decimal)

Returns the inverse sine of a number in radians. `Decimal` arguments are rounded to double precision before function application. Supported by SQL version 2015-10-8 and later.

Example: `asin(0)` = 0.0

Argument Type	Result
`Int`	`Decimal` (with double precision), the inverse sine of the argument. Imaginary results are returned as `Undefined`.
`Decimal`	`Decimal` (with double precision), the inverse sine of the argument. Imaginary results are returned as `Undefined`.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the inverse sine of the argument. If the string cannot be converted, the result is `Undefined`. Imaginary results are returned as `Undefined`.
Array	`Undefined`.
Object	`Undefined`.

Argument Type	Result
Null	`Undefined`.
Undefined	`Undefined`.

atan(Decimal)

Returns the inverse tangent of a number in radians. `Decimal` arguments are rounded to double precision before function application. Supported by SQL version 2015-10-8 and later.

Example: `atan(0)` = 0.0

Argument Type	Result
`Int`	`Decimal` (with double precision), the inverse tangent of the argument. Imaginary results are returned as `Undefined`.
`Decimal`	`Decimal` (with double precision), the inverse tangent of the argument. Imaginary results are returned as `Undefined`.
`Boolean`	`Undefined`.
`String`	`Decimal`, the inverse tangent of the argument. If the string cannot be converted, the result is `Undefined`. Imaginary results are returned as `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

atan2(Decimal, Decimal)

Returns the angle, in radians, between the positive x-axis and the (x, y) point defined in the two arguments. The angle is positive for counter-clockwise angles (upper half-plane, y > 0), and negative for clockwise angles (lower half-plane, y < 0). `Decimal` arguments are rounded to double precision before function application. Supported by SQL version 2015-10-8 and later.

Example: `atan2(1, 0)` = 1.5707963267948966

Argument Type	Argument Type	Result
`Int/Decimal`	`Int/Decimal`	`Decimal` (with doubl x-axis and the specifi
`Int/Decimal/String`	`Int/Decimal/String`	`Decimal`, the inverse string cannot be conv
Other Value	Other Value	`Undefined`.

aws_lambda(functionArn, inputJson)

Calls the specified Lambda function passing `inputJson` to the Lambda function and returns the JSON generated by the Lambda function.

Arguments

Argument	Description
`functionArn`	The ARN of the Lambda function to call. The Lambda function must return JSON data.
`inputJson`	The JSON input passed to the Lambda function.

You must grant AWS IoT `lambda:InvokeFunction` permissions to invoke the specified Lambda function. The following example shows how to grant the `lambda:InvokeFunction` permission using the AWS CLI:

```
aws lambda add-permission --function-name "function_name"
--region "region"
--principal iot.amazonaws.com
--source-arn arn:aws:iot:us-east-1:account_id:rule/rule_name
--source-account "account_id"
--statement-id "unique_id"
--action "lambda:InvokeFunction"
```

The following are the arguments for the `add-permission` command:

--function-name

Name of the Lambda function whose resource policy you are updating by adding a new permission.

--region

The AWS region of your account.

--principal

The principal who is getting the permission. This should be `iot.amazonaws.com` to allow AWS IoT permission to call a Lambda function.

--source-arn

The ARN of the rule. You can use the `get-topic-rule` CLI command to get the ARN of a rule.

--source-account

The AWS account where the rule is defined.

--statement-id

A unique statement identifier.

--action

The Lambda action you want to allow in this statement. In this case, we want to allow AWS IoT to invoke a Lambda function, so we specify `lambda:InvokeFunction`.

Note
If you add a permission for a AWS IoT principal without providing the source ARN, any AWS account that creates a rule with your Lambda action can trigger rules to invoke your Lambda function from AWS IoT

For more information, see Lambda Permission Model.

The following rule shows how to call the `aws_lambda` function:

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function",
 payload.inner.element).some.value as output FROM 'a/b'
```

`payload.inner.element` selects data from message published on topic 'a/b'.

`some.value` selects data from the output that is generated by the Lambda function.

> **Note**
> Rules Engine limits the execution duration of Lambda Functions. Lambda function calls from rules should be completed within 2000ms.

bitand(Int, Int)

Performs a bitwise AND on the bit representations of the two `Int`(-converted) arguments. Supported by SQL version 2015-10-8 and later.

Example: `bitand(13, 5)` = 5

Argument Type	Argument Type	Result
`Int`	`Int`	`Int`, a bitwise AND o
`Int/Decimal`	`Int/Decimal`	`Int`, a bitwise AND o numbers are rounded the arguments canno is `Undefined`.
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, a bitwise AND o are converted to `Dec` the nearest `Int`. If th `Undefined`.
Other Value	Other Value	`Undefined`.

bitor(Int, Int)

Performs a bitwise OR of the bit representations of the two arguments. Supported by SQL version 2015-10-8 and later.

Example: `bitor(8, 5)` = 13

Argument Type	Argument Type	Result
`Int`	`Int`	`Int`, the bitwise OR o
`Int/Decimal`	`Int/Decimal`	`Int`, the bitwise OR o numbers are rounded conversion fails, the r
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, the bitwise OR o converted to `Decima` `Int`. If the conversion

Argument Type	Argument Type	Result
Other Value	Other Value	`Undefined`.

bitxor(Int, Int)

Performs a bitwise XOR on the bit representations of the two `Int`(-converted) arguments. Supported by SQL version 2015-10-8 and later.

Example:`bitor(13, 5)` = 8

Argument Type	Argument Type	Result
`Int`	`Int`	`Int`, a bitwise XOR o
`Int/Decimal`	`Int/Decimal`	`Int`, a bitwise XOR o numbers are rounded
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, a bitwise XOR o converted to `Decima` `Int`. If any conversio
Other Value	Other Value	`Undefined`.

bitnot(Int)

Performs a bitwise NOT on the bit representations of the `Int`(-converted) argument. Supported by SQL version 2015-10-8 and later.

Example: `bitnot(13)` = 2

Argument Type	Result
`Int`	`Int`, a bitwise NOT of the argument.
`Decimal`	`Int`, a bitwise NOT of the argument. The `Decimal` value is rounded down to the nearest `Int`.
`String`	`Int`, a bitwise NOT of the argument. `Strings` are converted to `Decimals` and rounded down to the nearest `Int`. If any conversion fails, the result is `Undefined`.
Other Value	Other value.

cast()

Converts a value from one data type to another. Cast behaves mostly like the standard conversions, with the addition of the ability to cast numbers to/from `Booleans`. If AWS IoT cannot determine how to cast one type to another, the result is `Undefined`. Supported by SQL version 2015-10-8 and later. Format: cast(*`value`* as *`type`*).

Example:

```
cast(true as Decimal)  = 1.0
```

The following keywords may appear after "as" when calling `cast`:

Keyword	Result
`Decimal`	Casts value to `Decimal`.
Bool	Casts value to `Boolean`.
`Boolean`	Casts value to `Boolean`.
`String`	Casts value to `String`.
Nvarchar	Casts value to `String`.
Text	Casts value to `String`.
Ntext	Casts value to `String`.
varchar	Casts value to `String`.
`Int`	Casts value to `Int`.
`Int`	Casts value to `Int`.

Casting rules:

Cast to Decimal

Argument Type	Result
`Int`	A `Decimal` with no decimal point.
`Decimal`	The source value.
`Boolean`	true = 1.0, false = 0.0.
`String`	Will try to parse the string as a `Decimal`. We will attempt to parse strings matching the regex: ^-?\d+(\.\d+)?((?i)E-?\d+)?$. "0", "-1.2", "5E-12" are all examples of `String`s that would be converted automatically to `Decimal`s.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

Cast to Int

Argument Type	Result
`Int`	The source value.
`Decimal`	The source value, rounded down to the nearest `Int`.
`Boolean`	true = 1.0, false = 0.0.

Argument Type	Result
`String`	Will try to parse the string as a `Decimal`. We will attempt to parse strings matching the regex: ^-?\d+(\.\d+)?((?i)E-?\d+)?$. "0", "-1.2", "5E-12" are all examples of `String`s that would be converted automatically to `Decimal`s. Will attempt to convert the string to a `Decimal` and round down to the nearest `Int`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

Cast to `Boolean`

Argument Type	Result
`Int`	0 = False, any_nonzero_value = True.
`Decimal`	0 = False, any_nonzero_value = True.
`Boolean`	The source value.
`String`	"true" = True and "false" = False (case-insensitive). Other string values = `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

Cast to String

Argument Type	Result
`Int`	A string representation of the `Int`, in standard notation.
`Decimal`	A string representing the `Decimal` value, possibly in scientific notation.
`Boolean`	"true" or "false", all lowercase.
`String`	"true"=True and "false"=False (case-insensitive). Other string values = `Undefined`.
Array	The array serialized to JSON. The result string will be a comma-separated list enclosed in square brackets. `String`s are quoted. `Decimal`s, `Int`s, `Boolean`s are not.
Object	The object serialized to JSON. The JSON string will be a comma-separated list of key-value pairs and will

Argument Type	Result
	begin and end with curly braces. `Strings` are quoted. `Decimals`, `Ints`, `Booleans` and `Null` are not.
Null	`Undefined`.
Undefined	`Undefined`.

ceil(Decimal)

Rounds the given `Decimal` up to the nearest `Int`. Supported by SQL version 2015-10-8 and later.

Examples:

`ceil(1.2)` = 2

`ceil(11.2)` = -1

Argument Type	Result
`Int`	`Int`, the argument value.
`Decimal`	`Int`, the `Decimal` value rounded up to the nearest `Int`.
`String`	`Int`. The string is converted to `Decimal` and rounded up to the nearest `Int`. If the string cannot be converted to a `Decimal`, the result is `Undefined`.
Other Value	`Undefined`.

chr(String)

Returns the ASCII character that corresponds to the given `Int` argument. Supported by SQL version 2015-10-8 and later.

Examples:

`chr(65)` = "A".

`chr(49)` = "1".

Argument Type	Result
`Int`	The character corresponding to the specified ASCII value. If the argument is not a valid ASCII value, the result is `Undefined`.
`Decimal`	The character corresponding to the specified ASCII value. The `Decimal` argument is rounded down to the nearest `Int`. If the argument is not a valid ASCII value, the result is `Undefined`.
`Boolean`	`Undefined`.

Argument Type	Result
`String`	If the `String` can be converted to a `Decimal`, it is rounded down to the nearest `Int`. If the argument is not a valid ASCII value, the result is `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Other Value	`Undefined`.

clientid()

Returns the ID of the MQTT client sending the message, or `n/a` if the message wasn't sent over MQTT. Supported by SQL version 2015-10-8 and later.

Example:

`clientid()` = "123456789012"

concat()

Concatenates arrays or strings. This function accepts any number of arguments and returns a `String` or an `Array`. Supported by SQL version 2015-10-8 and later.

Examples:

`concat()` = `Undefined`.

`concat(1)` = "1".

`concat([1, 2, 3], 4)` = [1, 2, 3, 4].

`concat([1, 2, 3], "hello")` = [1, 2, 3, "hello"]

`concat("con", "cat")` = "concat"

`concat(1, "hello")` = "1hello"

`concat("he","is","man")` = "heisman"

`concat([1, 2, 3], "hello", [4, 5, 6])` = [1, 2, 3, "hello", 4, 5, 6]

Number of Arguments	Result
0	`Undefined`.
1	The argument is returned unmodified.
2+	If any argument is an `Array`, the result is a single array containing all of the arguments. If no arguments are `Arrays`, and at least one argument is a `String`, the result is the concatenation of the `String` representations of all the arguments. Arguments will be converted to `Strings` using the standard conversions listed above. .

cos(Decimal)

Returns the cosine of a number in radians. `Decimal` arguments are rounded to double precision before function application. Supported by SQL version 2015-10-8 and later.

Example:

`cos(0)` = 1.

Argument Type	Result
`Int`	`Decimal` (with double precision), the cosine of the argument. Imaginary results are returned as `Undefined`.
`Decimal`	`Decimal` (with double precision), the cosine of the argument. Imaginary results are returned as `Undefined`.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the cosine of the argument. If the string cannot be converted to a `Decimal`, the result is `Undefined`. Imaginary results are returned as `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

cosh(Decimal)

Returns the hyperbolic cosine of a number in radians. `Decimal` arguments are rounded to double precision before function application. Supported by SQL version 2015-10-8 and later.

Example: `cosh(2.3)` = 5.037220649268761.

Argument Type	Result
`Int`	`Decimal` (with double precision), the hyperbolic cosine of the argument. Imaginary results are returned as `Undefined`.
`Decimal`	`Decimal` (with double precision), the hyperbolic cosine of the argument. Imaginary results are returned as `Undefined`.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the hyperbolic cosine of the argument. If the string cannot be converted to a `Decimal`, the result is `Undefined`. Imaginary results are returned as `Undefined`.
Array	`Undefined`.

Argument Type	Result
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

encode(value, encodingScheme)

Use the `encode` function to encode the payload, which potentially might be non-JSON data, into its string representation based on the encoding scheme. Supported by SQL version 2016-03-23 and later.

value

Any of the valid expressions, as defined in AWS IoT SQL Reference (p. 182). In addition, you can specify * to encode the entire payload, regardless of whether it's in JSON format. If you supply an expression, the result of the evaluation will first be converted to a string before it is encoded.

encodingScheme

A literal string representing the encoding scheme you want to use. Currently, only `'base64'` is supported.

endswith(String, String)

Returns a `Boolean` indicating whether the first `String` argument ends with the second `String` argument. If either argument is `Null` or `Undefined`, the result is `Undefined`. Supported by SQL version 2015-10-8 and later.

Example: `endswith("cat","at")` = true.

argument Type 1	argument Type 2	Result
`String`	`String`	True if the first argun Otherwise, false.
Other Value	Other Value	Both arguments are c standard conversion r ends in the second ar either argument is `Nu` `Undefined`.

exp(Decimal)

Returns e raised to the `Decimal` argument. `Decimal` arguments are rounded to double precision before function application. Supported by SQL version 2015-10-8 and later.

Example: `exp(1)` = e.

Argument Type	Result
`Int`	`Decimal` (with double precision), e ^ argument.
`Decimal`	`Decimal` (with double precision), e ^ argument.

Argument Type	Result
`String`	`Decimal` (with double precision), e ^ argument. If the `String` cannot be converted to a `Decimal`, the result is `Undefined`.
Other Value	`Undefined`.

get

Extracts a value from a collection-like type (Array, String, Object). No conversion will be applied to the first argument. Conversion applies as documented in the table to the second argument. Supported by SQL version 2015-10-8 and later.

Examples:

`get(["a", "b", "c"], 1)` = "b"

`get({"a":"b"}, "a")` = "b"

`get("abc", 1)` = "b"

argument Type 1	argument Type 2	Result
Array	Any Type (converted to `Int`)	The item at the 0-bas by the second argum conversion is unsucce index is outside the b array.length), the resu
String	Any Type (converted to `Int`)	The character at the (by the second argum conversion is unsucce index is outside the b string.length), the res
Object	`String` (no conversion is applied)	The value stored in th corresponding to the argument.
Other Value	Any Value	`Undefined`.

get_thing_shadow(thingName, roleARN)

Returns the shadow of the specified thing. Supported by SQL version 2016-03-23 and later.

thingName

String: The name of the thing whose shadow you want to retrieve.

roleArn

String: A role ARN with `iot:GetThingShadow` permission.

Example:

```
SELECT * from 'a/b'

WHERE get_thing_shadow("MyThing","arn:aws:iam::123456789012:role/
AllowsThingShadowAccess") .state.reported.alarm = 'ON'
```

Hashing Functions

AWS IoT provides the following hashing functions:

- md2
- md5
- sha1
- sha224
- sha256
- sha384
- sha512

All hash functions expect one string argument. The result is the hashed value of that string. Standard string conversions apply to non-string arguments. All hash functions are supported by SQL version 2015-10-8 and later.

Examples:

`md2("hello")` = "a9046c73e00331af68917d3804f70655"

`md5("hello")` = "5d41402abc4b2a76b9719d911017c592"

indexof(String, String)

Returns the first index (0-based) of the second argument as a substring in the first argument. Both arguments are expected as strings. Arguments that are not strings are subjected to standard string conversion rules. This function does not apply to arrays, only to strings. Supported by SQL version 2015-10-8 and later.

Examples:

`indexof("abcd", "bc")` = 1

isNull()

Returns whether the argument is the `Null` value. Supported by SQL version 2015-10-8 and later.

Examples:

`isNull(5)` = false.

`isNull(Null)` = true.

Argument Type	Result
`Int`	false
`Decimal`	false
`Boolean`	false

Argument Type	Result
`String`	false
`Array`	false
`Object`	false
`Null`	true
`Undefined`	false

isUndefined()

Returns whether the argument is `Undefined`. Supported by SQL version 2015-10-8 and later.

Examples:

`isUndefined(5)` = false.

`isNull(floor([1,2,3]))` = true.

Argument Type	Result
`Int`	false
`Decimal`	false
`Boolean`	false
`String`	false
`Array`	false
`Object`	false
`Null`	false
`Undefined`	true

length(String)

Returns the number of characters in the provided string. Standard conversion rules apply to non-`String` arguments. Supported by SQL version 2015-10-8 and later.

Examples:

`length("hi")` = 2

`length(false)` = 5

ln(Decimal)

Returns the natural logarithm of the argument. `Decimal` arguments are rounded to double precision before function application. Supported by SQL version 2015-10-8 and later.

Example: `ln(e)` = 1.

Argument Type	Result
`Int`	`Decimal` (with double precision), the natural log of the argument.
`Decimal`	`Decimal` (with double precision), the natural log of the argument.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the natural log of the argument. If the string cannot be converted to a `Decimal` the result is `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

log(Decimal)

Returns the base 10 logarithm of the argument. `Decimal` arguments are rounded to double precision before function application. Supported by SQL version 2015-10-8 and later.

Example: `log(100)` = 2.0.

Argument Type	Result
`Int`	`Decimal` (with double precision), the base 10 log of the argument.
`Decimal`	`Decimal` (with double precision), the base 10 log of the argument.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the base 10 log of the argument. If the `String` cannot be converted to a `Decimal`, the result is `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

lower(String)

Returns the lowercase version of the given `String`. Non-string arguments are converted to `Strings` using the standard conversion rules. Supported by SQL version 2015-10-8 and later.

Examples:

`lower("HELLO")` = "hello".

`lower(["HELLO"])` = "[\"hello\"]".

lpad(String, Int)

Returns the `String` argument, padded on the left side with the number of spaces specified by the second argument. The `Int` argument must be between 0 and 1000. If the provided value is outside of this valid range, the argument will be set to the nearest valid value (0 or 1000). Supported by SQL version 2015-10-8 and later.

Examples:

`lpad("hello", 2)` = " hello".

`lpad(1, 3)` = " 1"

argument Type 1	argument Type 2	Result
`String`	`Int`	`String`, the provided with a number of spa
`String`	`Decimal`	The `Decimal` argume nearest `Int` and the the specified number
`String`	`String`	The second argument which is rounded dow `String` is padded wi the left. If the second an `Int`, the result is `U`
Other Value	`Int/Decimal/String`	The first value will be standard conversions be applied on that `St` result is `Undefined`.
Any Value	Other Value	`Undefined`.

ltrim(String)

Removes all leading whitespace (tabs and spaces) from the provided `String`. Supported by SQL version 2015-10-8 and later.

Example:

`Ltrim(" h i ")` = "hi ".

Argument Type	Result
`Int`	The `String` representation of the `Int` with all leading whitespace removed.
`Decimal`	The `String` representation of the `Decimal` with all leading whitespace removed.

Argument Type	Result
`Boolean`	The `String` representation of the boolean ("true" or "false") with all leading whitespace removed.
`String`	The argument with all leading whitespace removed.
Array	The `String` representation of the `Array` (using standard conversion rules) with all leading whitespace removed.
Object	The `String` representation of the Object (using standard conversion rules) with all leading whitespace removed.
Null	`Undefined`.
Undefined	`Undefined`.

machinelearning_predict(modelId)

Use the `machinelearning_predict` function to make predictions using the data from an MQTT message based on an Amazon Machine Learning (Amazon ML) model. Supported by SQL version 2015-10-8 and later. The arguments for the `machinelearning_predict` function are:

modelId

The ID of the model against which to run the prediction. The real-time endpoint of the model must be enabled.

roleArn

The IAM role that has a policy with `machinelearning:Predict` and `machinelearning:GetMLModel` permissions and allows access to the model against which the prediction is run.

record

The data to be passed into the Amazon ML Predict API. This should be represented as a single layer JSON object. If the record is a multi-level JSON object, the record will be flattened by serializing its values. For example, the following JSON:

```
{ "key1": {"innerKey1": "value1"}, "key2": 0}
```

would become:

```
{ "key1": "{\"innerKey1\": \"value1\"}", "key2": 0}
```

The function returns a JSON object with the following fields:

predictedLabel

The classification of the input based on the model.

details

Contains the following attributes:

PredictiveModelType

The model type. Valid values are REGRESSION, BINARY, MULTICLASS.

Algorithm

The algorithm used by Amazon ML to make predictions. The value must be SGD.

predictedScores

Contains the raw classification score corresponding to each label.

predictedValue

The value predicted by Amazon ML.

mod(Decimal, Decimal)

Returns the remainder of the division of the first argument by the second argument. Supported by SQL version 2015-10-8 and later. You can also use "%" as an infix operator for the same modulo functionality. Supported by SQL version 2015-10-8 and later.

Example: `mod(8, 3)` = 2.

Left Operand	Right Operand	Output
`Int`	`Int`	`Int`, the first argume
`Int/Decimal`	`Int/Decimal`	`Decimal`, the first ar
`String/Int/Decimal`	`String/Int/Decimal`	If all strings convert t argument modulo th `Undefined`.
Other Value	Other Value	`Undefined`.

nanvl(AnyValue, AnyValue)

Returns the first argument if it is a valid `Decimal`. Otherwise, the second argument is returned. Supported by SQL version 2015-10-8 and later.

Example: `Nanvl(8, 3)` = 8.

argument Type 1	argument Type 2	Output
Undefined	Any Value	The second argumen
Null	Any Value	The second argumen
`Decimal` (NaN)	Any Value	The second argumen
`Decimal` (not NaN)	Any Value	The first argument.
Other Value	Any Value	The first argument.

newuuid()

Returns a random 16-byte UUID. Supported by SQL version 2015-10-8 and later.

Example: `newuuid()` = `123a4567-b89c-12d3-e456-789012345000`

numbytes(String)

Returns the number of bytes in the UTF-8 encoding of the provided string. Standard conversion rules apply to non-`String` arguments. Supported by SQL version 2015-10-8 and later.

Examples:

`numbytes("hi")` = 2

`numbytes("€")` = 3

principal()

Returns the X.509 certificate fingerprint or thing name, depending on which endpoint, MQTT or HTTP, received the request. Supported by SQL version 2015-10-8 and later.

Example:

`principal()` = "ba67293af50bf2506f5f93469686da660c7c844e7b3950bfb16813e0d31e9373"

parse_time(String, Long, [String])

Use the `parse_time` function to format a timestamp into a human-readable date/time format. Supported by SQL version 2016-03-23 and later. The arguments for the `parse_time` function are:

pattern

(String) A date/time pattern which conforms to the ISO 8601 standard format. (Specifically, the function supports Joda-Time formats.)

timestamp

(Long) The time to be formatted in milliseconds since Unix epoch. See function timestamp() (p. 223).

timezone

(String) [Optional] The time zone of the formatted date/time. The default is "UTC". The function supports Joda-Time timezones

Examples:

When this message is published to the topic 'A/B', the payload `{"ts": "1970.01.01 AD at 21:46:40 CST"}` will be sent to the S3 bucket:

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "rule": {
        "awsIotSqlVersion": "2016-03-23",
        "sql": "SELECT parse_time("yyyy.MM.dd G 'at' HH:mm:ss z", 100000000, "America/
Belize" ) as ts FROM 'A/B'",

        "ruleDisabled": false,
        "actions": [
            {
                "s3": {
                    "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                    "bucketName": "BUCKET_NAME",
                    "key": "KEY_NAME"
                }
            }
        ],
        "ruleName": "RULE_NAME"
```

```
    }
}
```

When this message is published to the topic 'A/B', a payload similar to `{"ts": "2017.06.09 AD at 17:19:46 UTC"}` (but with the current date/time) will be sent to the S3 bucket:

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "rule": {
        "awsIotSqlVersion": "2016-03-23",
        "sql": "SELECT parse_time("yyyy.MM.dd G 'at' HH:mm:ss z", timestamp() ) as ts FROM
 'A/B'",
        "ruleDisabled": false,
        "actions": [
            {
                "s3": {
                    "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                    "bucketName": "BUCKET_NAME",
                    "key": "KEY_NAME"
                }
            }
        ],
        "ruleName": "RULE_NAME"
    }
}
```

`parse_time()` can also be used as a substitution template. For example, when this message is published to the topic 'A/B', the payload will be sent to the S3 bucket with key = "2017":

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "rule": {
        "awsIotSqlVersion": "2016-03-23",
        "sql": "SELECT * FROM 'A/B'",
        "ruleDisabled": false,
        "actions": [
            {
                "s3": {
                    "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                    "bucketName": BUCKET_NAME,
                    "key": "${parse_time("yyyy", timestamp(), "UTC")}"
                }
            }
        ],
        "ruleName": "RULE_NAME"
    }
}
```

power(Decimal, Decimal)

Returns the first argument raised to the second argument. `Decimal` arguments are rounded to double precision before function application. Supported by SQL version 2015-10-8 and later. Supported by SQL version 2015-10-8 and later.

Example: `power(2, 5)` = 32.0.

argument Type 1	argument Type 2	Output
`Int/Decimal`	`Int/Decimal`	A `Decimal` (with dou raised to the second a

argument Type 1	argument Type 2	Output
`Int/Decimal/String`	`Int/Decimal/String`	A `Decimal` (with dou raised to the second a are converted to `Dec` converted to `Decima`
Other Value	Other Value	`Undefined`.

rand()

Returns a pseudorandom, uniformly distributed double between 0.0 and 1.0. Supported by SQL version 2015-10-8 and later.

Example:

`rand()` = 0.8231909191640703

regexp_matches(String, String)

Returns whether the first argument contains a match for the second argument (regex).

Example:

`Regexp_matches("aaaa", "a{2,}")` = true.

`Regexp_matches("aaaa", "b")` = false.

First argument:

Argument Type	Result
`Int`	The `String` representation of the `Int`.
`Decimal`	The `String` representation of the `Decimal`.
`Boolean`	The `String` representation of the boolean ("true" or "false").
`String`	The `String`.
Array	The `String` representation of the `Array` (using standard conversion rules).
Object	The `String` representation of the Object (using standard conversion rules).
Null	`Undefined`.
Undefined	`Undefined`.

Second argument:

Must be a valid regex expression. Non-string types are converted to `String` using the standard conversion rules. Depending on the type, the resultant string may or may not be a valid regular expression. If the (converted) argument is not valid regex, the result is `Undefined`.

Third argument:

Must be a valid regex replacement string. (Can reference capture groups.) Non-string types will be converted to `String` using the standard conversion rules. If the (converted) argument is not a valid regex replacement string, the result is `Undefined`.

regexp_replace(String, String, String)

Replaces all occurrences of the second argument (regular expression) in the first argument with the third argument. Reference capture groups with "$". Supported by SQL version 2015-10-8 and later.

Example:

`Regexp_replace("abcd", "bc", "x")` = "axd".

`Regexp_replace("abcd", "b(.*)d", "$1")` = "ac".

First argument:

Argument Type	Result
`Int`	The `String` representation of the `Int`.
`Decimal`	The `String` representation of the `Decimal`.
`Boolean`	The `String` representation of the boolean ("true" or "false").
`String`	The source value.
Array	The `String` representation of the `Array` (using standard conversion rules).
Object	The `String` representation of the Object (using standard conversion rules).
Null	`Undefined`.
Undefined	`Undefined`.

Second argument:

Must be a valid regex expression. Non-string types are converted to `String`s using the standard conversion rules. Depending on the type, the resultant string may or may not be a valid regular expression. If the (converted) argument is not a valid regex expression, the result is `Undefined`.

Third argument:

Must be a valid regex replacement string. (Can reference capture groups.) Non-string types will be converted to `String`s using the standard conversion rules. If the (converted) argument is not a valid regex replacement string, the result is `Undefined`.

regexp_substr(String, String)

Finds the first match of the 2nd parameter (regex) in the first parameter. Reference capture groups with "$". Supported by SQL version 2015-10-8 and later.

Example:

`regexp_substr("hihihello", "hi")` => `"hi"`

`regexp_substr("hihihello", "(hi)*")` => `"hihi"`.

First argument:

Argument Type	Result
`Int`	The `String` representation of the `Int`.
`Decimal`	The `String` representation of the `Decimal`.
`Boolean`	The `String` representation of the boolean ("true" or "false").
`String`	The `String` argument.
Array	The `String` representation of the `Array` (using standard conversion rules).
Object	The `String` representation of the Object (using standard conversion rules).
Null	`Undefined`.
Undefined	`Undefined`.

Second argument:

Must be a valid regex expression. Non-string types are converted to `String`s using the standard conversion rules. Depending on the type, the resultant string may or may not be a valid regular expression. If the (converted) argument is not a valid regex expression, the result is `Undefined`.

Third argument:

Must be a valid regex replacement string. (Can reference capture groups.) Non-string types will be converted to `String` using the standard conversion rules. If the argument is not a valid regex replacement string, the result is `Undefined`.

rpad(String, Int)

Returns the string argument, padded on the right side with the number of spaces specified in the second argument. The `Int` argument must be between 0 and 1000. If the provided value is outside of this valid range, the argument will be set to the nearest valid value (0 or 1000). Supported by SQL version 2015-10-8 and later.

Examples:

`rpad("hello", 2)` = "hello ".

`rpad(1, 3)` = "1 ".

argument Type 1	argument Type 2	Result
`String`	`Int`	The `String` is padded on the right side with a number

argument Type 1	argument Type 2	Result
		of spaces equal to the provided `Int`.
`String`	`Decimal`	The `Decimal` argument will be rounded down to the nearest `Int` and the string is padded on the right side with a number of spaces equal to the provided `Int`.

argument Type 1	argument Type 2	Result
`String`	`String`	The second argument will be converted to a `Decimal`, which is rounded down to the nearest `Int`. The `String` is padded on the right side with a number of spaces equal to the `Int` value.

argument Type 1	argument Type 2	Result
Other Value	`Int/Decimal/String`	The first value will be converted to a `String` using the standard conversions, and the rpad function will be applied on that `String`. If it cannot be converted, the result is `Undefined`.
Any Value	Other Value	`Undefined`.

round(Decimal)

Rounds the given `Decimal` to the nearest `Int`. If the `Decimal` is equidistant from two `Int` values (for example, 0.5), the `Decimal` is rounded up. Supported by SQL version 2015-10-8 and later.

Example: `Round(1.2)` = 1.

`Round(1.5)` = 2.

`Round(1.7)` = 2.

`Round(-1.1)` = -1.

`Round(-1.5)` = -2.

Argument Type	Result
`Int`	The argument.
`Decimal`	`Decimal` is rounded down to the nearest `Int`.
`String`	`Decimal` is rounded down to the nearest `Int`. If the string cannot be converted to a `Decimal`, the result is `Undefined`.

Argument Type	Result
Other Value	`Undefined`.

rtrim(String)

Removes all trailing whitespace (tabs and spaces) from the provided `String`. Supported by SQL version 2015-10-8 and later.

Examples:

`rtrim(" h i ")` = " h i"

Argument Type	Result
`Int`	The `String` representation of the `Int`.
`Decimal`	The `String` representation of the `Decimal`.
`Boolean`	The `String` representation of the boolean ("true" or "false").
Array	The `String` representation of the `Array` (using standard conversion rules).
Object	The `String` representation of the Object (using standard conversion rules).
Null	`Undefined`.
Undefined	`Undefined`

sign(Decimal)

Returns the sign of the given number. When the sign of the argument is positive, 1 is returned. When the sign of the argument is negative, -1 is returned. If the argument is 0, 0 is returned. Supported by SQL version 2015-10-8 and later.

Examples:

`sign(-7)` = -1.

`sign(0)` = 0.

`sign(13)` = 1.

Argument Type	Result
`Int`	`Int`, the sign of the `Int` value.
`Decimal`	`Int`, the sign of the `Decimal` value.
`String`	`Int`, the sign of the `Decimal` value. The string is converted to a `Decimal` value, and the sign of the `Decimal` value is returned. If the `String` cannot be converted to a `Decimal`, the result is `Undefined`. Supported by SQL version 2015-10-8 and later.

Argument Type	Result
Other Value	`Undefined`.

sin(Decimal)

Returns the sine of a number in radians. `Decimal` arguments are rounded to double precision before function application. Supported by SQL version 2015-10-8 and later.

Example: `sin(0)` = 0.0

Argument Type	Result
`Int`	`Decimal` (with double precision), the sine of the argument.
`Decimal`	`Decimal` (with double precision), the sine of the argument.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the sine of the argument. If the string cannot be converted to a `Decimal`, the result is `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
`Undefined`	`Undefined`.

sinh(Decimal)

Returns the hyperbolic sine of a number. `Decimal` values are rounded to double precision before function application. The result is a `Decimal` value of double precision. Supported by SQL version 2015-10-8 and later.

Example: `sinh(2.3)` = 4.936961805545957

Argument Type	Result
`Int`	`Decimal` (with double precision), the hyperbolic sine of the argument.
`Decimal`	`Decimal` (with double precision), the hyperbolic sine of the argument.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the hyperbolic sine of the argument. If the string cannot be converted to a `Decimal`, the result is `Undefined`.
Array	`Undefined`.

Argument Type	Result
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

substring(String, Int [, Int])

Expects a `String` followed by one or two `Int` values. For a `String` and a single `Int` argument, this function returns the substring of the provided `String` from the provided `Int` index (0-based, inclusive) to the end of the `String`. For a `String` and two `Int` arguments, this function returns the substring of the provided `String` from the first `Int` index argument (0-based, inclusive) to the second `Int` index argument (0-based, exclusive). Indices that are less than zero will be set to zero. Indices that are greater than the `String` length will be set to the `String` length. For the three argument version, if the first index is greater than (or equal to) the second index, the result is the empty `String`.

If the arguments provided are not (*`String`*, *`Int`*), or (*`String`*, *`Int`*, *`Int`*), the standard conversions will be applied to the arguments to attempt to convert them into the correct types. If the types cannot be converted, the result of the function is `Undefined`. Supported by SQL version 2015-10-8 and later.

Examples:

`substring("012345", 0)` = "012345".

`substring("012345", 2)` = "2345".

`substring("012345", 2.745)` = "2345".

`substring(123, 2)` = "3".

`substring("012345", -1)` = "012345".

`substring(true, 1.2)` = "rue".

`substring(false, -2.411E247)` = "false".

`substring("012345", 1, 3)` = "12".

`substring("012345", -50, 50)` = "012345".

`substring("012345", 3, 1)` = "".

sqrt(Decimal)

Returns the square root of a number. `Decimal` arguments are rounded to double precision before function application. Supported by SQL version 2015-10-8 and later.

Example: `sqrt(9)` = 3.0.

Argument Type	Result
`Int`	The square root of the argument.
`Decimal`	The square root of the argument.

Argument Type	Result
`Boolean`	`Undefined`.
`String`	The square root of the argument. If the string cannot be converted to a `Decimal`, the result is `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

startswith(String, String)

Returns `Boolean`, whether the first string argument starts with the second string argument. If either argument is `Null` or `Undefined`, the result is `Undefined`. Supported by SQL version 2015-10-8 and later.

Example:

`startswith("ranger","ran")` = true

argument Type 1	argument Type 2	Result
`String`	`String`	Whether the first stri
Other Value	Other Value	Both arguments will b the standard conversi string starts with the `Null` or `Undefined`,

tan(Decimal)

Returns the tangent of a number in radians. `Decimal` values are rounded to double precision before function application. Supported by SQL version 2015-10-8 and later.

Example: `tan(3)` = -0.1425465430742778

Argument Type	Result
`Int`	`Decimal` (with double precision), the tangent of the argument.
`Decimal`	`Decimal` (with double precision), the tangent of the argument.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the tangent of the argument. If the string cannot be converted to a `Decimal`, the result is `Undefined`.
Array	`Undefined`.

Argument Type	Result
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

tanh(Decimal)

Returns the hyperbolic tangent of a number in radians. `Decimal` values are rounded to double precision before function application. Supported by SQL version 2015-10-8 and later.

Example: `tanh(2.3)` = 0.9800963962661914

Argument Type	Result
`Int`	`Decimal` (with double precision), the hyperbolic tangent of the argument.
`Decimal`	`Decimal` (with double precision), the hyperbolic tangent of the argument.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the hyperbolic tangent of the argument. If the string cannot be converted to a `Decimal`, the result is `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

timestamp()

Returns the current timestamp in milliseconds from 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970, as observed by the AWS IoT rules engine. Supported by SQL version 2015-10-8 and later.

Example: `timestamp()` = `1481825251155`

topic(Decimal)

Returns the topic to which the message that triggered the rule was sent. If no parameter is specified, the entire topic is returned. The `Decimal` parameter is used to specify a specific, one-based topic segment. For the topic `foo/bar/baz`, topic(1) will return `foo`, topic(2) will return `bar`, and so on. Supported by SQL version 2015-10-8 and later.

Examples:

`topic()` = "things/myThings/thingOne"

`topic(1)` = "things"

traceid()

Returns the trace ID (UUID) of the MQTT message, or `Undefined` if the message wasn't sent over MQTT. Supported by SQL version 2015-10-8 and later.

Example:

`traceid()` = "12345678-1234-1234-1234-123456789012"

trunc(Decimal, Int)

Truncates the first argument to the number of `Decimal` places specified by the second argument. If the second argument is less than zero, it will be set to zero. If the second argument is greater than 34, it will be set to 34. Trailing zeroes are stripped from the result. Supported by SQL version 2015-10-8 and later.

Examples:

`trunc(2.3, 0)` = 2.

`trunc(2.3123, 2` = 2.31.

`trunc(2.888, 2)` = 2.88.

`(2.00, 5)` = 2.

argument Type 1	argument Type 2	Result
`Int`	`Int`	The source value.
`Int/Decimal`	`Int/Decimal`	The first argument is by the second argum `Int`, will be rounded
`Int/Decimal/String`	The first argument is truncated to the length described by the second argument. The second argument, if not an `Int`, will be rounded down to the nearest `Int`. `Strings` are converted to `Decimal` values. If the string conversion fails, the result is `Undefined`.	
Other Value	`Undefined`.	

trim(String)

Removes all leading and trailing whitespace from the provided `String`. Supported by SQL version 2015-10-8 and later.

Example:

`Trim(" hi ")` = "hi"

Argument Type	Result
`Int`	The `String` representation of the `Int` with all leading and trailing whitespace removed.
`Decimal`	The `String` representation of the `Decimal` with all leading and trailing whitespace removed.

Argument Type	Result
`Boolean`	The `String` representation of the `Boolean` ("true" or "false") with all leading and trailing whitespace removed.
`String`	The `String` with all leading and trailing whitespace removed.
Array	The `String` representation of the `Array` using standard conversion rules.
Object	The `String` representation of the Object using standard conversion rules.
Null	`Undefined`.
Undefined	`Undefined`.

upper(String)

Returns the uppercase version of the given `String`. Non-`String` arguments are converted to `String` using the standard conversion rules. Supported by SQL version 2015-10-8 and later.

Examples:

`upper("hello")` = "HELLO"

`upper(["hello"])` = "[\"HELLO\"]"

SELECT Clause

The AWS IoT SELECT clause is essentially the same as the ANSI SQL SELECT clause, with some minor differences.

You can use the SELECT clause to extract information from incoming MQTT messages. `SELECT *` can be used to retrieve the entire incoming message payload. For example:

```
Incoming payload published on topic 'a/b': {"color":"red", "temperature":50}
SQL statement: SELECT * FROM 'a/b'
Outgoing payload: {"color":"red", "temperature":50}
```

If the payload is a JSON object, you can reference keys in the object. Your outgoing payload will contain the key-value pair. For example:

```
Incoming payload published on topic 'a/b': {"color":"red", "temperature":50}
SQL statement: SELECT color FROM 'a/b'
Outgoing payload: {"color":"red"}
```

You can use the AS keyword to rename keys. For example:

```
Incoming payload published on topic 'a/b':{"color":"red", "temperature":50}
SQL:SELECT color AS my_color FROM 'a/b'
Outgoing payload: {"my_color":"red"}
```

You can select multiple items by separating them with a comma. For example:

```
Incoming payload published on topic 'a/b': {"color":"red", "temperature":50}
SQL: SELECT color as my_color, temperature as farenheit FROM 'a/b'
Outgoing payload: {"my_color":"red","farenheit":50}
```

You can select multiple items including '*' to add items to the incoming payload. For example:

```
Incoming payload published on topic 'a/b': {"color":"red", "temperature:50}
SQL: SELECT *, 15 as speed FROM 'a/b'
Outgoing payload: {"color":"red", "temperature:50, speed:15}"
```

You can use the `"VALUE"` keyword to produce outgoing payloads that are not JSON objects. You may only select one item. For example:

```
Incoming payload published on topic 'a/b': {"color":"red", "temperature":50}
SQL: SELECT VALUE color FROM 'a/b'
Outgoing payload: "red"
```

You can use `'.'` syntax to drill into nested JSON objects in the incoming payload. For example:

```
Incoming payload published on topic 'a/b': {"color":{"red":255,"green":0,"blue":0},
 "temperature":50}
SQL: SELECT color.red as red_value FROM 'a/b'
Outgoing payload: {"red_value":255}
```

You can use functions (see Functions (p. 192)) to transform the incoming payload. Parentheses can be used for grouping. For example:

```
Incoming payload published on topic 'a/b': {"color":"red", "temperature":50}
SQL: SELECT (temperature - 32) * 5 / 9 AS celsius, upper(color) as my_color FROM 'a/b'
Outgoing payload: {"celsius":10,"my_color":"RED"}
```

Working with Binary Payloads

When the message payload should be handled as raw binary data (rather than a JSON object), you can use the * operator to refer to it in a `SELECT` clause.

These rules must be followed to use * to refer to the message payload as raw binary data:

1. The SQL statement and templates must not refer to JSON names, other than *.
2. The SELECT statement must have * as the only item, or must have only functions, for example:

```
SELECT * FROM 'a/b'
```

```
SELECT encode(*, 'base64') AS data, timestamp() AS ts FROM 'a/b'
```

Binary Payload Examples

The following `SELECT` clause can be used with binary payloads because it doesn't refer to any JSON names.

```
SELECT * FROM 'a/b'
```

The following `SELECT` can not be used with binary payloads because it refers to `device_type` in the WHERE clause.

```
SELECT * FROM 'a/b' WHERE device_type = 'thermostat'
```

The following `SELECT` can not be used with binary payloads because it violates rule #2.

```
SELECT *, timestamp() AS timestamp FROM 'a/b'
```

The following `SELECT` can be used with binary payloads because it doesn't violate either rule #1 or #2.

```
SELECT * FROM 'a/b' WHERE timestamp() % 12 = 0
```

The following AWS IoT rule can not be used with payloads because it violates rule #1.

```
{
    "sql": "SELECT * FROM 'a/b'"
    "actions": [{
        "republish": {
            "topic":"device/${device_id}"
         }
     }]
}
```

FROM Clause

The FROM clause subscribes your rule to a topic or topic filter. A topic filter allows you to subscribe to a group of similar topics.

Example:

Incoming payload published on topic `'a/b'`: `{temperature: 50}`

Incoming payload published on topic `'a/c'`: `{temperature: 50}`

SQL: `"SELECT temperature AS t FROM 'a/b'"`.

The rule is subscribed to `'a/b'`, so the incoming payload is passed to the rule, and the outgoing payload (passed to the rule actions) is: `{t: 50}`. The rule is not subscribed to `'a/c'`, so the rule is not triggered for the message published on `'a/c'`.

You can use the # wildcard character to match any subpath in a topic filter:

Example:

Incoming payload published on topic `'a/b'`: `{temperature: 50}`.

Incoming payload published on topic `'a/c'`: `{temperature: 60}`.

Incoming payload published on topic `'a/e/f'`: `{temperature: 70}`.

Incoming payload published on topic `'b/x'`: `{temperature: 80}`.

SQL: `"SELECT temperature AS t FROM 'a/#'"`.

The rule is subscribed to any topic beginning with `'a'`, so it is executed three times, sending outgoing payloads of `{t: 50} (for a/b), {t: 60}` (for a/c), and `{t: 70}` (for a/e/f) to its actions. It is not subscribed to `'b/x'`, so the rule will not be triggered for the {temperature: 80} message.

You can use the '+' character to match any one particular path element:

Example:

Incoming payload published on topic `'a/b'`: `{temperature: 50}`.

Incoming payload published on topic `'a/c'`: `{temperature: 60}`.

Incoming payload published on topic `'a/e/f'`: `{temperature: 70}`.

Incoming payload published on topic `'b/x'`: `{temperature: 80}`.

SQL: `"SELECT temperature AS t FROM 'a/+'"`.

The rule is subscribed to all topics with two path elements where the first element is `'a'`. The rule is executed for the messages sent to `'a/b'` and `'a/c'`, but not `'a/e/f'` or `'b/x'`.

You can use functions and operators in the WHERE clause. In the WHERE clause, you cannot reference any aliases created with the AS keyword in the SELECT. (The WHERE clause is evaluated first, to determine if the SELECT clause is evaluated.)

WHERE Clause

The WHERE clause determines if a rule is evaluated for a message sent to an MQTT topic to which the rule is subscribed. If the WHERE clause evaluates to true, the rule is evaluated. Otherwise, the rule is not evaluated.

Example:

Incoming payload published on `a/b`: `{"color":"red", "temperature":40}`.

SQL: `SELECT color AS my_color FROM 'a/b' WHERE temperature > 50 AND color <> 'red'`.

In this case, the rule would not be evaluated; there would be no outgoing payload; and rules actions would not be triggered.

You can use functions and operators in the WHERE clause. However, you cannot reference any aliases created with the AS keyword in the SELECT. (The WHERE clause is evaluated first, to determine if SELECT is evaluated.)

Literals

You can directly specify literal objects in the SELECT and WHERE clauses of your rule SQL, which can be useful for passing information.

> **Note**
> Literals are only available when using SQL Version 2016-03-23 or newer.

JSON object syntax is used (key-value pairs, comma-separated, where keys are strings and values are JSON values, wrapped in curly brackets {}). For example:

Incoming payload published on topic `a/b`: `"{lat_long: [47.606,-122.332]}"`

SQL statement: `SELECT {'latitude': get(lat_long, 0),'longitude':get(lat_long, 1)} as lat_long FROM 'a/b'`

The resulting outgoing payload would be: `{'latitude':47.606,'longitude':-122.332}`.

You can also directly specify arrays in the SELECT and WHERE clauses of your rule SQL, which allows you to group information. JSON syntax is used (wrap comma-separated items in square brackets [] to create an array literal). For example:

Incoming payload published on topic `a/b`: `{lat: 47.696, long: -122.332}`

SQL statement: `SELECT [lat,long] as lat_long FROM 'a/b'`

The resulting output payload would be: `{"lat_long": [47.606,-122.332]}`.

Case Statements

Case statements can be used for branching execution, like a switch statement, or if/else statements.

Syntax:

```
CASE v WHEN t[1] THEN r[1]
  WHEN t[2] THEN r[2] ...
  WHEN t[n] THEN r[n]
  ELSE r[e] END
```

The expression `v` is evaluated and matched for equality against each `t[i]` expression. If a match is found, the corresponding `r[i]` expression becomes the result of the case statement. If there is more than one possible match, the first match is selected. If there are no matches, the else statement's `re` is used as the result. If there is no match and no else statement, the result of the case statement is `Undefined`. For example:

Incoming payload published on topic `a/b`: `{"color":"yellow"}`

SQL statement: `SELECT CASE color WHEN 'green' THEN 'go' WHEN 'yellow' THEN 'caution' WHEN 'red' THEN 'stop' ELSE 'you are not at a stop light' END as instructions FROM 'a/b'`

The resulting output payload would be: `{"instructions":"caution"}`.

Case statements require at least one WHEN clause. An ELSE clause is not required.

> **Note**
> If v is `Undefined`, the result of the case statement is `Undefined`.

JSON Extensions

You can use the following extensions to ANSI SQL syntax to make it easier to work with nested JSON objects.

"." Operator

This operator accesses members in embedded JSON objects and functions identically to ANSI SQL and JavaScript. For example:

```
SELECT foo.bar AS bar.baz FROM 'a/b'
```

`*` Operator

This functions in the same way as the `*` wildcard in ANSI SQL. It's used in the SELECT clause only and creates a new JSON object containing the message data. If the message payload is not in JSON format, `*` returns the entire message payload as raw bytes. For example:

```
SELECT * FROM 'a/b'
```

Applying a Function to an Attribute Value

The following is an example JSON payload that could be published by a device:

```
{
    "deviceid" : "iot123",
    "temp" : 54.98,
    "humidity" : 32.43,
    "coords" : {
        "latitude" : 47.615694,
        "longitude" : -122.3359976
    }
}
```

The following example applies a function to an attribute value in a JSON payload:

```
SELECT temp, md5(deviceid) AS hashed_id FROM topic/#
```

The result of this query is the following JSON object:

```
{
   "temp": 54.98,
   "hashed_id": "e37f81fb397e595c4aeb5645b8cbbbd1"
}
```

Substitution Templates

You can use a substitution template to augment the JSON data returned when a rule is triggered and AWS IoT performs an action. The syntax for a substitution template is `${`*expression*`}`, where *expression* can be any expression supported by AWS IoT in SELECT or WHERE clauses. For more information about supported expressions, see AWS IoT SQL Reference (p. 182).

Substitution templates appear in the SELECT clause within a rule:

```
{
    "sql": "SELECT *, topic() AS topic FROM 'my/iot/topic'",
    "ruleDisabled": false,
    "actions": [{
        "republish": {
            "topic": "${topic()}/republish",
            "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
        }
    }]
}
```

If this rule is triggered by the following JSON:

```
{
    "deviceid" : "iot123",
    "temp" : 54.98,
    "humidity" : 32.43,
    "coords" : {
        "latitude" : 47.615694,
        "longitude" : -122.3359976
    }
```

```
}
```

Here is the output of the rule:

```
{
    "coords":{
        "longitude":-122.3359976,
        "latitude":47.615694
    },
    "humidity":32.43,
    "temp":54.98,
    "deviceid":"iot123",
    "topic":"my/iot/topic"
}
```

Thing Shadows for AWS IoT

A *thing shadow* (sometimes referred to as a device shadow) is a JSON document that is used to store and retrieve current state information for a thing (device, app, and so on). The Thing Shadows service maintains a thing shadow for each thing you connect to AWS IoT. You can use thing shadows to get and set the state of a thing over MQTT or HTTP, regardless of whether the thing is connected to the Internet. Each thing shadow is uniquely identified by its name.

Contents

Thing Shadows Data Flow

The Thing Shadows service acts as an intermediary, allowing devices and applications to retrieve and update thing shadows.

To illustrate how devices and applications communicate with the Thing Shadows service, this section walks you through the use of the AWS IoT MQTT client and the AWS CLI to simulate communication between an internet-connected light bulb, an application, and the Thing Shadows service.

The Thing Shadows service uses MQTT topics to facilitate communication between applications and devices. To see how this works, use the AWS IoT MQTT client to subscribe to the following MQTT topics with QoS 1:

$aws/things/myLightBulb/shadow/update/accepted

The Thing Shadows service sends messages to this topic when an update is successfully made to the thing shadow.

$aws/things/myLightBulb/shadow/update/rejected

The Thing Shadows service sends messages to this topic when an update to the thing shadow is rejected.

$aws/things/myLightBulb/shadow/update/delta

The Thing Shadows service sends messages to this topic when a difference is detected between the reported and desired sections of the thing shadow. For more information, see /update/delta (p. 256).

$aws/things/myLightBulb/shadow/get/accepted

The Thing Shadows service sends messages to this topic when a request for the thing shadow is made successfully.

$aws/things/myLightBulb/shadow/get/rejected

The Thing Shadows service sends messages to this topic when a request for the thing shadow is rejected.

$aws/things/myLightBulb/shadow/delete/accepted

The Thing Shadows service sends messages to this topic when the thing shadow is deleted.

$aws/things/myLightBulb/shadow/delete/rejected

The Thing Shadows service sends messages to this topic when a request to delete the thing shadow is rejected.

$aws/things/myLightBulb/shadow/update/documents

The Thing Shadows service publishes a state document to this topic whenever an update to the thing shadow is successfully performed.

To learn more about all of the MQTT topics used by the Thing Shadows service, see Thing Shadow MQTT Topics (p. 253).

Note
We recommend that you subscribe to the `.../rejected` topics to see any errors sent by the Thing Shadows service.

When the light bulb comes online, it sends its current state to the Thing Shadows service by sending an MQTT message to the `$aws/things/myLightBulb/shadow/update` topic.

To simulate this, use the AWS IoT MQTT client to publish the following message to the `$aws/things/myLightbulb/shadow/update` topic:

```
{
    "state": {
        "reported": {
            "color": "red"
        }
    }
}
```

This message sets the color of the light bulb to "red."

The Thing Shadows service responds by sending the following message to the `$aws/things/myLightBulb/shadow/update/accepted` topic:

```
{
  "messageNumber": 4,
  "payload": {
    "state": {
      "reported": {
        "color": "red"
      }
    },
    "metadata": {
      "reported": {
        "color": {
          "timestamp": 1469564492
        }
      }
    },
    "version": 1,
    "timestamp": 1469564492
  },
  "qos": 0,
  "timestamp": 1469564492848,
  "topic": "$aws/things/myLightBulb/shadow/update/accepted"
}
```

This message indicates the Thing Shadows service received the UPDATE request and updated the thing shadow. If the thing shadow doesn't exist, it is created. Otherwise, the thing shadow is updated with the data in the message. If you don't see a message published to `$aws/things/myLightBulb/shadow/update/accepted`, check the subscription to `$aws/things/myLightBulb/shadow/update/rejected` to see any error messages.

In addition, the Thing Shadows service publishes the following message to the `$aws/things/myLightBulb/shadow/update/documents` topic.

```
{
    "previous":null,
    "current":{
        "state":{
            "reported":{
                "color":"red"
            }
         },
         "metadata":{
            "reported":{
                "color":{
                    "timestamp":1483467764
                }
            }
         },
         "version":1
     },
     "timestamp":1483467764
}
```

Messages are published to the `/update/documents` topic whenever an update to the thing shadow is successfully performed. For more information of the contents of messages published to this topic, see Thing Shadow MQTT Topics (p. 253).

An application that interacts with the light bulb comes online and requests the light bulb's current state. The application sends an empty message to the `$aws/things/myLightBulb/shadow/get` topic. To simulate this, use the AWS IoT MQTT client to publish an empty message ("") to the `$aws/things/myLightBulb/shadow/get` topic.

The Thing Shadows service responds by publishing the requested thing shadow to the `$aws/things/myLightBulb/shadow/get/accepted` topic:

```
{
  "messageNumber": 1,
  "payload": {
    "state": {
      "reported": {
        "color": "red"
      }
    },
    "metadata": {
      "reported": {
        "color": {
          "timestamp": 1469564492
        }
      }
    },
    "version": 1,
    "timestamp": 1469564571
  },
  "qos": 0,
  "timestamp": 1469564571533,
  "topic": "$aws/things/myLightBulb/shadow/get/accepted"
```

```
}
```

If you don't see a message on the `$aws/things/myLightBulb/shadow/get/accepted` topic, check the `$aws/things/myLightBulb/shadow/get/rejected` topic for any error messages.

The application displays this information to the user, and the user requests a change to the light bulb's color (from red to green). To do this, the application publishes a message on the `$aws/things/myLightBulb/shadow/update` topic:

```
{
    "state": {
        "desired": {
            "color": "green"
        }
    }
}
```

To simulate this, use the AWS IoT MQTT client to publish the preceding message to the `$aws/things/myLightBulb/shadow/update` topic.

The Thing Shadows service responds by sending a message to the `$aws/things/myLightBulb/shadow/update/accepted` topic:

```
{
  "messageNumber": 5,
  "payload": {
    "state": {
      "desired": {
        "color": "green"
      }
    },
    "metadata": {
      "desired": {
        "color": {
          "timestamp": 1469564658
        }
      }
    },
    "version": 2,
    "timestamp": 1469564658
  },
  "qos": 0,
  "timestamp": 1469564658286,
  "topic": "$aws/things/myLightBulb/shadow/update/accepted"
}
```

and to the `$aws/things/myLightBulb/shadow/update/delta` topic:

```
{
  "messageNumber": 1,
  "payload": {
    "version": 2,
    "timestamp": 1469564658,
    "state": {
      "color": "green"
    },
    "metadata": {
      "color": {
        "timestamp": 1469564658
      }
    }
```

```
  },
  "qos": 0,
  "timestamp": 1469564658309,
  "topic": "$aws/things/myLightBulb/shadow/update/delta"
}
```

The Thing Shadow service publishes a message to this topic when it accepts a thing shadow update and the resulting thing shadow contains different values for desired and reported states.

The Thing Shadow service also publishes a message to the `$aws/things/myLightBulb/shadow/update/documents` topic:

```
{
    "previous":{
        "state":{
            "reported":{
                "color":"red"
            }
        },
        "metadata":{
            "reported":{
                "color":{
                    "timestamp":1483467764
                }
            }
        },
        "version":1
    },
    "current":{
        "state":{
            "desired":{
                "color":"green"
            },
            "reported":{
                "color":"red"
            }
        },
        "metadata":{
            "desired":{
                "color":{
                    "timestamp":1483468612
                }
            },
            "reported":{
                "color":{
                    "timestamp":1483467764
                }
            }
       },
       "version":2
   },
   "timestamp":1483468612
}
```

The light bulb is subscribed to the `$aws/things/myLightBulb/shadow/update/delta` topic, so it receives the message, changes its color, and publishes its new state. To simulate this, use the AWS IoT MQTT client to publish the following message to the `$aws/things/myLightbulb/shadow/update` topic to update the shadow state:

```
{
    "state":{
        "reported":{
            "color":"green"
```

```
        },
        "desired":null}
    }
}
```

In response, the Thing Shadows service sends a message to the `$aws/things/myLightBulb/shadow/update/accepted` topic:

```
{
  "messageNumber": 6,
  "payload": {
    "state": {
      "reported": {
        "color": "green"
      },
      "desired": null
    },
    "metadata": {
      "reported": {
        "color": {
          "timestamp": 1469564801
        }
      },
      "desired": {
        "timestamp": 1469564801
      }
    },
    "version": 3,
    "timestamp": 1469564801
  },
  "qos": 0,
  "timestamp": 1469564801673,
  "topic": "$aws/things/myLightBulb/shadow/update/accepted"
}
```

and to the `$aws/things/myLightBulb/shadow/update/documents` topic:

```
{
    "previous":{
    "state":{
        "reported":{
            "color":"red"
        }
    },
    "metadata":{
         "reported":{
              "color":{
                  "timestamp":1483470355
              }
          }
      },
      "version":3
    },
    "current":{
        "state":{
            "reported":{
                "color":"green"
            }
        },
        "metadata":{
            "reported":{
                "color":{
                    "timestamp":1483470364
```

```
                }
            }
        },
        "version":4
    },
    "timestamp":1483470364
}
```

The app requests the current state from the Thing Shadows service and displays the most recent state data. To simulate this, run the following command:

```
aws iot-data get-thing-shadow --thing-name "myLightBulb" "output.txt" && cat "output.txt"
```

Note
On Windows, omit the `&& cat "output.txt"`, which displays the contents of output.txt to the console. You can open the file in Notepad or any text editor to see the contents of the thing shadow.

The Thing Shadows service returns the thing shadow document:

```
{
    "state":{
        "reported":{
            "color":"green"
        }
    },
    "metadata":{
        "reported":{
            "color":{
                "timestamp":1469564801
            }
        }
    },
    "version":3,
    "timestamp":1469564864}
```

To delete the thing shadow, publish an empty message to the `$aws/things/myLightBulb/shadow/delete` topic. AWS IoT responds by publishing a message to the `$aws/things/myLightBulb/shadow/delete/accepted` topic:

```
{
  "version" : 1,
  "timestamp" : 1488565234
}
```

Detecting a Thing Is Connected

To determine if a device is currently connected, include a connected setting in the thing shadow and use an MQTT Last Will and Testament (LWT) message that sets the connected setting to `false` if a device is disconnected due to error.

Note
Currently, LWT messages sent to AWS IoT reserved topics (topics that begin with $) are ignored by the AWS IoT Shadows service, but are still processed by subscribed clients and by the AWS IoT rules engine. If you want the AWS IoT Shadows service to receive LWT messages, register an LWT message to a non-reserved topic and create a rule that republishes the message on the reserved topic. The following example shows how to create a republish rule that listens for a messages from the `my/things/myLightBulb/update` topic and republishes it to `$aws/things/myLightBulb/shadow/update`.

```
{
    "rule": {
    "ruleDisabled": false,
    "sql": "SELECT * FROM 'my/things/myLightBulb/update'",
    "description": "Turn my/things/ into $aws/things/",
    "actions": [{
        "republish": {
            "topic": "$$aws/things/myLightBulb/shadow/update",
            "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish"
        }
    }]
    }
}
```

When a device connects, it registers an LWT that sets the connected setting to `false`:

```
{
    "state": {
        "reported":
        {
            "connected":"false"
        }
    }
}
```

It also publishes a message on its update topic (`$aws/things/myLightBulb/shadow/update`), setting its connected state to `true`:

```
{
     "state": {
        "reported":
        {
            "connected":"true"
        }
    }
}
```

When the device disconnects gracefully, it publishes a message on its update topic and sets its connected state to `false`:

```
{
    "state": {
        "reported":{
            "connected":"false"
        }
    }
}
```

If the device disconnects due to an error, its LWT message is posted automatically to the update topic.

Thing Shadows Documents

The Thing Shadows service respects all rules of the JSON specification. Values, objects, and arrays are stored in the thing shadow document.

Contents

Document Properties

A thing shadow document has the following properties:

`state`

`desired`

The desired state of the thing. Applications can write to this portion of the document to update the state of a thing without having to directly connect to a thing.

`reported`

The reported state of the thing. Things write to this portion of the document to report their new state. Applications read this portion of the document to determine the state of a thing.

`metadata`

Information about the data stored in the `state` section of the document. This includes timestamps, in Epoch time, for each attribute in the `state` section, which enables you to determine when they were updated.

`timestamp`

Indicates when the message was transmitted by AWS IoT. By using the timestamp in the message and the timestamps for individual attributes in the `desired` or `reported` section, a thing can determine how old an updated item is, even if it doesn't feature an internal clock.

`clientToken`

A string unique to the device that enables you to associate responses with requests in an MQTT environment.

`version`

The document version. Every time the document is updated, this version number is incremented. Used to ensure the version of the document being updated is the most recent.

For more information, see Thing Shadow Document Syntax (p. 260).

Versioning of a Thing Shadow

The Thing Shadows service supports versioning on every update message (both request and response), which means that with every update of a thing shadow, the version of the JSON document is incremented. This ensures two things:

- A client can receive an error if it attempts to overwrite a shadow using an older version number. The client is informed it must resync before it can update a thing shadow.
- A client can decide not to act on a received message if the message has a lower version than the version stored by the client.

In some cases, a client might bypass version matching by not submitting a version.

Client Token

You can use a client token with MQTT-based messaging to verify the same client token is contained in a request and request response. This ensures the response and request are associated.

Example Document

Here is an example thing shadow document:

```
{
    "state" : {
        "desired" : {
          "color" : "RED",
          "sequence" : [ "RED", "GREEN", "BLUE" ]
        },
        "reported" : {
          "color" : "GREEN"
        }
    },
    "metadata" : {
        "desired" : {
            "color" : {
                "timestamp" : 12345
            },
            "sequence" : {
                "timestamp" : 12345
            }
        },
        "reported" : {
            "color" : {
                "timestamp" : 12345
            }
        }
    },
    "version" : 10,
    "clientToken" : "UniqueClientToken",
    "timestamp": 123456789
}
```

Empty Sections

A thing shadow document contains a `desired` section only if it has a desired state. For example, the following is a valid state document with no `desired` section:

```
{
    "reported" : { "temp": 55 }
}
```

The `reported` section can also be empty:

```
{
    "desired" : { "color" : "RED" }
}
```

If an update causes the `desired` or `reported` sections to become null, the section is removed from the document. To remove the `desired` section from a document (in response, for example, to a device updating its state), set the desired section to `null`:

```
{
    "state": {
        "reported": {
            "color": "red"
        },
        "desired": null
    }
}
```

It is also possible a thing shadow document will not contain `desired` or `reported` sections. In that case, the shadow document is empty. For example, this is a valid document:

```
{
}
```

Arrays

Thing shadows support arrays, but treat them as normal values in that an update to an array replaces the whole array. It is not possible to update part of an array.

Initial state:

```
{
    "desired" : { "colors" : ["RED", "GREEN", "BLUE" ] }
}
```

Update:

```
{
    "desired" : { "colors" : ["RED"] }
}
```

Final state:

```
{
    "desired" : { "colors" : ["RED"] }
}
```

Arrays can't have null values. For example, the following array is not valid and will be rejected.

```
{
    "desired" : {
        "colors" : [ null, "RED", "GREEN" ]
    }
}
```

Using Thing Shadows

AWS IoT provides three methods for working with thing shadows:

`UPDATE`

Creates a thing shadow if it doesn't exist, or updates the content of a thing shadow with the data provided in the request. The data is stored with timestamp information to indicate when it was last

updated. Messages are sent to all subscribers with the difference between `desired` or `reported` state (delta). Things or apps that receive a message can perform an action based on the difference between `desired` or `reported` states. For example, a device can update its state to the desired state, or an app can update its UI to show the change in the device's state.

`GET`

Retrieves the latest state stored in the thing shadow (for example, during start-up of a device to retrieve configuration and the last state of operation). This method returns the full JSON document, including metadata.

`DELETE`

Deletes a thing shadow, including all of its content. This removes the JSON document from the data store. You can't restore a thing shadow you deleted, but you can create a new thing shadow with the same name.

Protocol Support

These methods are supported through both MQTT and a RESTful API over HTTPS. Because MQTT is a publish/subscribe communication model, AWS IoT implements a set of reserved topics. Things or applications subscribe to these topics before publishing on a request topic in order to implement a request–response behavior. For more information, see Thing Shadow MQTT Topics (p. 253) and Thing Shadow RESTful API (p. 251).

Updating a Thing Shadow

You can update a thing shadow by using the UpdateThingShadow (p. 252) RESTful API or by publishing to the /update (p. 254) topic. Updates affect only the fields specified in the request.

Initial state:

```
{
    "state": {
        "reported" : {
            "color" : { "r" :255, "g": 255, "b": 0 }
        }
    }
}
```

An update message is sent:

```
{
    "state": {
        "desired" : {
            "color" : { "r" : 10 },
            "engine" : "ON"
        }
    }
}
```

The device receives the `desired` state on the `/update/delta` topic that is triggered by the previous `/update` message and then executes the desired changes. When finished, the device should confirm its updated state through the `reported` section in the thing shadow JSON document.

Final state:

```
{
```

```
    "state": {
        "reported" : {
            "color" : {  "r" : 10, "g" : 255, "b": 0 },
            "engine" : "ON"
        }
    }
}
```

Retrieving a Thing Shadow Document

You can retrieve a thing shadow by using the GetThingShadow (p. 251) RESTful API or by subscribing and publishing to the /get (p. 257) topic. This retrieves the entire document plus the delta between the `desired` or `reported` states.

Example document:

```
{
    "state": {
        "desired": {
            "lights": {
                "color": "RED"
            },
            "engine": "ON"
        },
        "reported": {
            "lights": {
                "color": "GREEN"
            },
            "engine": "ON"
        }
    },
    "metadata": {
        "desired": {
            "lights": {
                "color": {
                    "timestamp": 123456
                },
                "engine": {
                    "timestamp": 123456
                }
            }
        },
        "reported": {
            "lights": {
                "color": {
                    "timestamp": 789012
                }
            },
            "engine": {
                "timestamp": 789012
            }
        },
        "version": 10,
        "timestamp": 123456789
    }
}
```

Response:

```
{
    "state": {
        "desired": {
```

```
            "lights": {
                "color": "RED"
            },
            "engine": "ON"
        },
        "reported": {
            "lights": {
                "color": "GREEN"
            },
            "engine": "ON"
        },
        "delta": {
            "lights": {
                "color": "RED"
            }
        }
    },
    "metadata": {
        "desired": {
            "lights": {
                "color": {
                    "timestamp": 123456
                },
             }
             "engine": {
                "timestamp": 123456
            }
        },
        "reported": {
            "lights": {
                "color": {
                    "timestamp": 789012
                }
            },
            "engine": {
                "timestamp": 789012
            }
        },
        "delta": {
            "lights": {
                "color": {
                    "timestamp": 123456
                }
            }
        }
    },
    "version": 10,
    "timestamp": 123456789
}
```

Optimistic Locking

You can use the state document version to ensure you are updating the most recent version of a thing shadow document. When you supply a version with an update request, the service rejects the request with an HTTP 409 conflict response code if the current version of the state document does not match the version supplied.

For example:

Initial document:

```
{
    "state" : {
```

```
        "desired" : { "colors" : ["RED", "GREEN", "BLUE" ] }
    },
    "version" : 10
}
```

Update: (version doesn't match; request will be rejected)

```
{
    "state": {
        "desired": {
            "colors": [
                "BLUE"
            ]
        }
    },
    "version": 9
}
```

Result:

```
409 Conflict
```

Update: (version matches; this request will be accepted)

```
{
    "state": {
        "desired": {
            "colors": [
                "BLUE"
            ]
        }
    },
    "version": 10
}
```

Final state:

```
{
    "state": {
        "desired": {
            "colors": [
                "BLUE"
            ]
        }
    },
    "version": 11
}
```

Deleting Data

You can delete data from a thing shadow by publishing to the /update (p. 254) topic, setting the fields to be deleted to null. Any field with a value of `null` is removed from the document.

Initial state:

```
{
    "state": {
        "desired" : {
            "lights": { "color": "RED" },
```

```
            "engine" : "ON"
        },
        "reported" : {
            "lights" : { "color": "GREEN"  },
            "engine" : "OFF"
        }
    }
}
```

An update message is sent:

```
{
    "state": {
        "desired": null,
        "reported": {
            "engine": null
        }
    }
}
```

Final state:

```
{
    "state": {
        "reported" : {
            "lights" : { "color" : "GREEN" }
        }
    }
}
```

You can delete all data from a thing shadow by setting its state to `null`. For example, sending the following message deletes all of the state data, but the thing shadow remains.

```
{
    "state": null
}
```

The thing shadow still exists even if its state is `null`. The version of the thing shadow is incremented when the next update occurs.

Deleting a Thing Shadow

You can delete a thing shadow document by using the DeleteThingShadow (p. 253) RESTful API or by publishing to the /delete (p. 258) topic.

> **Note**
> Deleting a thing shadow does not delete the thing. Deleting a thing does not delete the thing shadow.

Initial state:

```
{
    "state": {
        "desired" : {
            "lights": { "color": "RED" },
            "engine" : "ON"
        },
        "reported" : {
            "lights" : { "color": "GREEN"  },
            "engine" : "OFF"
```

```
        }
    }
}
```

An empty message is published to the /delete topic.

Final state:

```
HTTP 404 - resource not found
```

Delta State

Delta state is a virtual type of state that contains the difference between the `desired` and `reported` states. Fields in the `desired` section that are not in the `reported` section are included in the delta. Fields that are in the `reported` section and not in the `desired` section are not included in the delta. The delta contains metadata, and its values are equal to the metadata in the `desired` field. For example:

```
{
    "state": {
        "desired": {
            "color": "RED",
            "state": "STOP"
        },
        "reported": {
            "color": "GREEN",
            "engine": "ON"
        },
        "delta": {
            "color": "RED",
            "state": "STOP"
        }
    },
    "metadata": {
        "desired": {
            "color": {
                "timestamp": 12345
            },
            "state": {
                "timestamp": 12345
            },
            "reported": {
                "color": {
                    "timestamp": 12345
                },
                "engine": {
                    "timestamp": 12345
                }
            },
            "delta": {
                "color": {
                    "timestamp": 12345
                },
                "state": {
                    "timestamp": 12345
                }
            }
        },
        "version": 17,
        "timestamp": 123456789
    }
}
```

When nested objects differ, the delta contains the path all the way to the root.

```
{
    "state": {
        "desired": {
            "lights": {
                "color": {
                    "r": 255,
                    "g": 255,
                    "b": 255
                }
            }
        },
        "reported": {
            "lights": {
                "color": {
                    "r": 255,
                    "g": 0,
                    "b": 255
                }
            }
        },
        "delta": {
            "lights": {
                "color": {
                    "g": 255
                }
            }
        }
    },
    "version": 18,
    "timestamp": 123456789
}
```

The Thing Shadows service calculates the delta by iterating through each field in the `desired` state and comparing it to the `reported` state.

Arrays are treated like values. If an array in the `desired` section doesn't match the array in the `reported` section, then the entire desired array is copied into the delta.

Observing State Changes

When a thing shadow is updated, messages are published on two MQTT topics:

- $aws/things/*thing-name*/shadow/update/accepted
- $aws/things/*thing-name*/shadow/update/delta

The message sent to the `update/delta` topic is intended for the thing whose state is being updated. This message contains only the difference between the `desired` and `reported` sections of the thing shadow document. Upon receiving this message, the thing decides whether to make the requested change. If the thing's state is changed, it publishes its new current state to the `$aws/things/thing-name/shadow/update` topic.

Devices and applications can subscribe to either of these topics to be notified when the state of the document has changed.

Here is an example of that flow:

1. Device reports state.
2. The system updates the state document in its persistent data store.

3. The system publishes a delta message, which contains only the delta and is targeted at the subscribed devices. Devices should subscribe to this topic to receive updates.
4. The thing shadow publishes an accepted message, which contains the entire received document, including metadata. Applications should subscribe to this topic to receive updates.

Message Order

There is no guarantee that messages from the AWS IoT service will arrive at the device in any specific order.

Initial state document:

```
{
    "state" : {
        "reported" : { "color" : "blue" }
    },
    "version" : 10,
    "timestamp": 123456777
}
```

Update 1:

```
{
    "state": { "desired" : { "color" : "RED" } },
    "version": 10,
    "timestamp": 123456777
}
```

Update 2:

```
{
    "state": { "desired" : { "color" : "GREEN" } },
    "version": 11 ,
    "timestamp": 123456778
}
```

Final state document:

```
{
    "state": {
        "reported": { "color" : "GREEN" }
    },
    "version": 12,
    "timestamp": 123456779
}
```

This results in two delta messages:

```
{
    "state": {
        "color": "RED"
    },
    "version": 11,
    "timestamp": 123456778
}
```

```
{
```

```
    "state": { "color" : "GREEN" },
    "version": 12,
    "timestamp": 123456779
}
```

The device might receive these messages out of order. Because the state in these messages is cumulative, a device can safely discard any messages that contain a version number older than the one it is tracking. If the device receives the delta for version 12 before version 11, it can safely discard the version 11 message.

Trim Thing Shadow Messages

To reduce the size of thing shadow messages sent to your device, define a rule that selects only the fields your device needs and republishes the message on an MQTT topic to which your device is listening.

The rule is specified in JSON and should look like the following:

```
{
    "sql": "SELECT state, version FROM '$aws/things/+/shadow/update/delta'",
    "ruleDisabled": false,
    "actions": [{
        "republish": {
            "topic": "${topic(2)}/delta",
            "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
        }
    }]
}
```

The SELECT statement determines which fields from the message will be republished to the specified topic. A "+" wild card is used to match all thing shadow names. The rule specifies that all matching messages should be republished to the specified topic. In this case, the `"topic()"` function is used to specify the topic on which to republish. `topic(2)` evaluates to the thing name in the original topic. For more information about creating rules, see Rules.

Thing Shadow RESTful API

A thing shadow exposes the following URI for updating state information:

```
https://endpoint/things/thingName/shadow
```

The endpoint is specific to your AWS account. To retrieve your endpoint, use the describe-endpoint command. The format of the endpoint is as follows:

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

API Actions

- GetThingShadow (p. 251)
- UpdateThingShadow (p. 252)
- DeleteThingShadow (p. 253)

GetThingShadow

Gets the thing shadow for the specified thing.

The response state document includes the delta between the `desired` and `reported` states.

Request

The request includes the standard HTTP headers plus the following URI:

```
HTTP GET https://endpoint/things/thingName/shadow
```

Response

Upon success, the response includes the standard HTTP headers plus the following code and body:

```
HTTP 200
BODY: response state document
```

For more information, see Example Response State Document (p. 260).

Authorization

Retrieving a thing shadow requires a policy that allows the caller to perform the `iot:GetThingShadow` action. The Thing Shadows service accepts two forms of authentication: Signature Version 4 with IAM credentials or TLS mutual authentication with a client certificate.

The following is an example policy that allows a caller to retrieve a thing shadow:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "iot:GetThingShadow",
        "Resource": ["arn:aws:iot:region:account:thing/thing"]
    }]
}
```

UpdateThingShadow

Updates the thing shadow for the specified thing.

Updates affect only the fields specified in the request state document. Any field with a value of `null` is removed from the thing shadow.

Request

The request includes the standard HTTP headers plus the following URI and body:

```
HTTP POST https://endpoint/things/thingName/shadow
BODY: request state document
```

For more information, see Example Request State Document (p. 260).

Response

Upon success, the response includes the standard HTTP headers plus the following code and body:

```
HTTP 200
BODY: response state document
```

For more information, see Example Response State Document (p. 260).

Authorization

Updating a thing shadow requires a policy that allows the caller to perform the `iot:UpdateThingShadow` action. The Thing Shadows service accepts two forms of authentication: Signature Version 4 with IAM credentials or TLS mutual authentication with a client certificate.

The following is an example policy that allows a caller to update a thing shadow:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "iot:UpdateThingShadow",
        "Resource": ["arn:aws:iot:region:account:thing/thing"]
    }]
}
```

DeleteThingShadow

Deletes the thing shadow for the specified thing.

Request

The request includes the standard HTTP headers plus the following URI:

```
HTTP DELETE https://endpoint/things/thingName/shadow
```

Response

Upon success, the response includes the standard HTTP headers plus the following code and body:

```
HTTP 200
BODY: Empty response state document
```

Authorization

Deleting a thing shadow requires a policy that allows the caller to perform the `iot:DeleteThingShadow` action. The Thing Shadows service accepts two forms of authentication: Signature Version 4 with IAM credentials or TLS mutual authentication with a client certificate.

The following is an example policy that allows a caller to delete a thing shadow:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "iot:DeleteThingShadow",
        "Resource": ["arn:aws:iot:region:account:thing/thing"]
    }]
}
```

Thing Shadow MQTT Topics

The Thing Shadows service uses reserved MQTT topics to enable applications and things to get, update, or delete the state information for a thing (thing shadow). The names of these topics start with $aws/

things/*thingName*/shadow. Publishing and subscribing on thing shadow topics requires topic-based authorization. AWS IoT reserves the right to add new topics to the existing topic structure. For this reason, we recommend that you avoid wild card subscriptions to shadow topics. For example, avoid subscribing to topic filters like `$aws/things/thingName/shadow/#` because the number of topics that match this topic filter might increase as AWS IoT introduces new shadow topics. For examples of the messages published on these topics see Thing Shadows Data Flow (p. 232).

The following are the MQTT topics used for interacting with thing shadows.

Topics

/update

Publish a request state document to this topic to update the thing shadow:

```
$aws/things/thingName/shadow/update
```

A client attempting to update the state of a thing would send a JSON request state document like this:

```
{
    "state" : {
        "desired" : {
            "color" : "red",
            "power" : "on"
         }
     }
}
```

A thing updating its thing shadow would send a JSON request state document like this:

```
{
    "state" : {
        "reported" : {
            "color" : "red",
            "power" : "on"
         }
     }
}
```

AWS IoT responds by publishing to either /update/accepted (p. 255) or /update/rejected (p. 256).

For more information, see Request State Documents (p. 260).

Example Policy

The following is an example of the required policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": ["arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/
update"]
     }]
}
```

/update/accepted

AWS IoT publishes a response state document to this topic when it accepts a change for the thing shadow:

```
$aws/things/thingName/shadow/update/accepted
```

For more information, see Response State Documents (p. 260).

Example Policy

The following is an example of the required policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
update/accepted"]
    }]
}
```

/update/documents

AWS IoT publishes a state document to this topic whenever an update to the shadow is successfully performed:

```
$aws/things/thingName/shadow/update/documents
```

The JSON document will contain two primary nodes: `previous` and `current`. The `previous` node will contain the contents of the full shadow document before the update was performed while `current` will contain the full shadow document after the update is successfully applied. When the thing shadow is updated (created) for the first time, the `previous` node will contain `null`.

Example Policy

The following is an example of the required policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
update/documents"]
    }]
}
```

/update/rejected

AWS IoT publishes an error response document to this topic when it rejects a change for the thing shadow:

```
$aws/things/thingName/shadow/update/rejected
```

For more information, see Error Response Documents (p. 261).

Example Policy

The following is an example of the required policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
update/rejected"]
    }]
}
```

/update/delta

AWS IoT publishes a response state document to this topic when it accepts a change for the thing shadow and the request state document contains different values for `desired` and `reported` states:

```
$aws/things/thingName/shadow/update/delta
```

For more information, see Response State Documents (p. 260).

Publishing Details

- A message published on `update/delta` includes only the desired attributes that differ between the `desired` and `reported` sections. It contains all of these attributes, regardless of whether these attributes were contained in the current update message or were already stored in AWS IoT. Attributes that do not differ between the `desired` and `reported` sections are not included.
- If an attribute is in the `reported` section but has no equivalent in the `desired` section, it is not included.

- If an attribute is in the `desired` section but has no equivalent in the `reported` section, it is included.
- If an attribute is deleted from the `reported` section but still exists in the `desired` section, it is included.

Example Policy

The following is an example of the required policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
update/delta"]
    }]
}
```

/get

Publish an empty message to this topic to get the thing shadow:

```
$aws/things/thingName/shadow/get
```

AWS IoT responds by publishing to either /get/accepted (p. 257) or /get/rejected (p. 258).

Example Policy

The following is an example of the required policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iot:Publish"
        ],
        "Resource": ["arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/get"]
    }]
}
```

/get/accepted

AWS IoT publishes a response state document to this topic when returning the thing shadow:

```
$aws/things/thingName/shadow/get/accepted
```

For more information, see Response State Documents (p. 260).

Example Policy

The following is an example of the required policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
get/accepted"]
    }]
}
```

/get/rejected

AWS IoT publishes an error response document to this topic when it can't return the thing shadow:

```
$aws/things/thingName/shadow/get/rejected
```

For more information, see Error Response Documents (p. 261).

Example Policy

The following is an example of the required policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Action": [
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
get/rejected"]
    }]
}
```

/delete

To delete a thing shadow, publish an empty message to the delete topic:

```
$aws/things/thingName/shadow/delete
```

The content of the message is ignored.

AWS IoT responds by publishing to either /delete/accepted (p. 259) or /delete/rejected (p. 259).

Example Policy

The following is an example of the required policy:

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

```
        "Action": [
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource": ["arn:aws:iot:region:account:topic filter/$aws/things/thingName/shadow/
delete"]
    }]
}
```

/delete/accepted

AWS IoT publishes a message to this topic when a thing shadow is deleted:

```
$aws/things/thingName/shadow/delete/accepted
```

Example Policy

The following is an example of the required policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
delete/accepted"]
    }]
}
```

/delete/rejected

AWS IoT publishes an error response document to this topic when it can't delete the thing shadow:

```
$aws/things/thingName/shadow/delete/rejected
```

For more information, see Error Response Documents (p. 261).

Example Policy

The following is an example of the required policy:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
delete/rejected"]
    }]
}
```

Thing Shadow Document Syntax

The Thing Shadows service uses the following documents in UPDATE, GET, and DELETE operations using the RESTful API (p. 251) or MQTT Pub/Sub Messages (p. 253). For more information, see Thing Shadows Documents (p. 239).

Examples

- Request State Documents (p. 260)
- Response State Documents (p. 260)
- Error Response Documents (p. 261)

Request State Documents

Request state documents have the following format:

```
{
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer1,
            "attribute2": "string1",
            ...
            "attributeN": boolean1
        }
    }
    "clientToken": "token",
    "version": version
}
```

- `state` — Updates affect only the fields specified.
- `clientToken` — If used, you can verify that the request and response contain the same client token.
- `version` — If used, the Thing Shadows service processes the update only if the specified version matches the latest version it has.

Response State Documents

Response state documents have the following format:

```
{
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer1,
            "attribute2": "string1",
            ...
```

```
            "attributeN": boolean1
        },
        "delta": {
            "attribute3": integerX,
            "attribute5": "stringY"
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        },
        "reported": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "timestamp": timestamp,
    "clientToken": "token",
    "version": version
}
```

- `state`
  - `reported` — Only present if a thing reported any data in the `reported` section and contains only fields that were in the request state document.
  - `desired` — Only present if a thing reported any data in the `desired` section and contains only fields that were in the request state document.
- `metadata` — Contains the timestamps for each attribute in the `desired` and `reported` sections so that you can determine when the state was updated.
- `timestamp` — The Epoch date and time the response was generated by AWS IoT.
- `clientToken` — Present only if a client token was used when publishing valid JSON to the `/update` topic.
- `version` — The current version of the document for the thing shadow shared in AWS IoT. It is increased by one over the previous version of the document.

Error Response Documents

Error response documents have the following format:

```
{
    "code": error-code,
    "message": "error-message",
    "timestamp": timestamp,
```

```
    "clientToken": "token"
}
```

- `code` — An HTTP response code that indicates the type of error.
- `message` — A text message that provides additional information.
- `timestamp` — The date and time the response was generated by AWS IoT.
- `clientToken` — Present only if a client token was used when publishing valid JSON to the `/update` topic.

For more information, see Thing Shadow Error Messages (p. 262).

Thing Shadow Error Messages

The Thing Shadows service publishes a message on the error topic (over MQTT) when an attempt to change the state document fails. This message is only emitted as a response to a publish request on one of the reserved $aws topics. If the client updates the document using the REST API, then it receives the HTTP error code as part of its response, and no MQTT error messages are emitted.

HTTP Error Code	Error Messages
400 (Bad Request)	• Invalid JSON • Missing required node: state • State node must be an object • Desired node must be an object • Reported node must be an object • Invalid version • Invalid clientToken • JSON contains too many levels of nesting; maximum is 6 • State contains an invalid node
401 (Unauthorized)	• Unauthorized
403 (Forbidden)	• Forbidden
404 (Not Found)	• Thing not found
409 (Conflict)	• Version conflict
413 (Payload Too Large)	• The payload exceeds the maximum size allowed
415 (Unsupported Media Type)	• Unsupported documented encoding; supported encoding is UTF-8
429 (Too Many Requests)	• The Thing Shadow service will generate this error message when there are more than 10 in-flight requests.
500 (Internal Server Error)	• Internal service failure

Jobs

AWS IoT Jobs is a service that allows you to define a set of *jobs* — remote operations that are sent to and executed on one or more devices connected to AWS IoT. For example, you can define a job that instructs a set of devices to download and install application or firmware updates, reboot, rotate certificates, or perform remote troubleshooting operations.

To create a job, you make a *job document* which is a description of the remote operations to be performed, and you specify a list of *targets* that should perform the operations. The targets can be individual things, thing groups (p. 99) or both.

AWS IoT Jobs sends a message to inform the targets that a job is available. The target starts the *execution* of the job by downloading the job document, performing the operations it specifies, and reporting its progress to AWS IoT. The Jobs service provides commands to track the progress of a job on a specific target and for all the targets of the job.

Managing Jobs

Jobs are created and managed using the Jobs HTTPS API. For more information, see Job Management and Control API (p. 277).

Continuous Jobs

By default, a job is sent to all targets that you specify when you create the job. After those targets complete the job (or report that they are unable to do so), the job is complete. However,you can make a job *continuous* by setting an optional parameter when you create the job. A continuous job is one that continues to run and is executed when a change is detected in a target. For example, a job will run on a device when the thing representing the device is added to a target group, even after the job was completed by all things originally in the group. A continuous job can be used to onboard or upgrade devices as they are added to a group.

Configuring Rollouts

configuring rollouts

When you create a job you can specify how quickly targets are notified of a pending job execution. This allows you to create a staged rollout to better manage updates, reboots, and other operations.

more info (1)

The following field can be added to the `CreateJob` request:

```
"jobExecutionRolloutConfig": {
    "maximumPerMinute": "integer"
}
```

Job Documents

job documents

Job documents are UTF-8 encoded JSON documents that contain the information your devices need to perform a job. A job document will most likely contain one or more URLs where the device can

download an update or some other data. The job document itself can be stored in an Amazon S3 bucket, or be included inline with the command that creates the job.

more info (2)

To allow a device secure, time-limited access to data beyond that included in the job document itself, you can use presigned Amazon S3 URLs. You can place your data in an Amazon S3 bucket and add a placeholder link to the data in the job document. When the Jobs service receives a request for the job document, it parses the job document looking for placeholder links and it replaces them with presigned Amazon S3 URLs.

The placeholder link is of the following form:

```
${aws:iot:s3-presigned-url:https://s3.amazonaws.com/bucket/key}
```

where *bucket* is your bucket name and *key* is the object in the bucket to which you are linking.

When creating a job that uses presigned Amazon S3 URLs, you must provide an IAM role ARN that grants permission to download files from the Amazon S3 bucket where the data or updates are stored. The role must also grant permission for AWS IoT to assume the role.

To grant Jobs permission to assume your role:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Roles**.
3. Search for your role and choose it.
4. Choose the **Trust Relationships** tab.
5. Choose the **Edit Trust Relationship** button.
6. On the **Edit Trust Relationship** page, replace the policy document with the following JSON:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iot.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

7. Choose **Update Trust Policy**

You can optionally specify a timeout for the presigned URL. For more information, see CreateJob (p. 284).

Create Jobs

CreateJob

You use the `CreateJob` command to create a job. The job is queued for execution on the targets (things or thing groups) that you specify. To create a job, you need a job document that can be included in the body of the request or as a link to an Amazon S3 document. If the job includes downloading files using presigned Amazon S3 URLs, you need an IAM role ARN that has permission to download the file and grants permission to AWS IoT to assume the role.

more info (3)

If you have a file called job-document.json stored in an Amazon S3 bucket called jobBucket and the role with permission to download files from Amazon S3 is called S3DownloadRole, the CLI command to create a job would look like this:

```
aws iot create-job  \
  --job-id 010  \
  --targets arn:aws:iot:us-east-1:123456789012:thing/thingOne  \
  --document-source https://s3.amazonaws.com/jobBucket/job-document.json  \
  --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/
S3DownloadRole\", \"expiresInSec\":3600}"
```

If you want to specify the job document inline, use the `--document` parameter instead of the `--document-source` parameter.

A job is sent to and executed on targets in the order they appear in the `--targets` list. For example, if the targets list is:

```
[ arn:aws:iot:us-east-1:123456789012:thing/thingOne,
    arn:aws:iot:us-east-1:123456789012:thinggroup/thinggroupOne,
    arn:aws:iot:us-east-1:123456789012:thing/thingTwo,
    arn:aws:iot:us-east-1:123456789012:thinggroup/thinggroupTwo ]
```

then the job will be executed on `thingOne`, followed by the things in `thinggroupOne` then `thingTwo` and finally by the things in `thinggroupTwo`.

Note
Job documents that are specified as Amazon S3 files are retrieved at the time you create the job. Changing the contents of the Amazon S3 file you used as the source of your job document after you have created the job does not change what is sent to the targets of the job.

Cancel a Job

CancelJob

You use the the `CancelJob` command to cancel a job. Cancelling a job will stop AWS IoT rolling out any new job executions for the job. It will also cancel any job executions that have already been `QUEUED`. IoT will leave any job executions in an `IN_PROGRESS` or terminal status untouched because the device is already executing the job or has already completed it.

more info (4)

```
aws iot cancel-job --job-id 010
```

The command displays no output.

Job executions that are `QUEUED` will be canceled. Job executions that are `IN_PROGRESS` or in a terminal state are not canceled. The status of a canceled job or of one of its job execution is eventually consistent — IoT will stop scheduling new job executions and not present `QUEUED` job executions for that job to devices as soon as possible. But changing the status of a job execution to `CANCELED` may take some time depending on the number of devices and other factors.

Get a Job Document

GetJobDocument

You use the `GetJobDocument` command to retrieve a job document for a job.

more info (5)

```
aws iot get-job-document --job-id 010
```

The command returns the job document for the specified job:

```
{
    "document": "{\n\t\"operation\":\"install\",\n\t\"url\":\"http://amazon.com/
firmWareUpate-01\",\n\t\"data\":\"${aws:iot:s3-presigned-url:https://s3.amazonaws.com/
job-test-bucket/datafile}\"\n}"
}
```

Note
When you retrieve a job document using this command, any placeholder URLs are not replaced by presigned Amazon S3 URLs. The placeholder URLs are replaced by presigned Amazon S3 URLs in the job document that is returned when a device calls the GetPendingJobExecutions (p. 294) MQTT API.

Tracking Jobs

List Jobs

ListJobs

You use the `ListJobs` command to get a list of all jobs in your AWS account. Note that job data and job execution data will be purged after 90 days.

more info (6)

```
aws iot list-jobs
```

The command lists all jobs in your account sorted by the job status:

```
{
    "jobs": [
        {
            "status": "IN_PROGRESS",
            "lastUpdatedAt": 1486687079.743,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/013",
            "createdAt": 1486687079.743,
            "targetSelection": "SNAPSHOT",
```

```
            "jobId": "013"
        },
        {
            "status": "COMPLETED",
            "lastUpdatedAt": 1486685868.444,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/012",
            "createdAt": 1486685868.444,
            "completedAt": 148668789.690,
            "targetSelection": "SNAPSHOT",
            "jobId": "012"
        },
        {
            "status": "CANCELED",
            "lastUpdatedAt": 1486678850.575,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/011",
            "createdAt": 1486678850.575,
            "targetSelection": "SNAPSHOT",
            "jobId": "011"
        }
    ]
}
```

Describe a Job

DescribeJob

You use the `DescribeJob` command to get the status of a specific job.

more info (7)

```
$ aws iot describe-job --job-id 010
```

The command returns the status of the specified job:

```
{
    "documentSource": "https://s3.amazonaws.com/job-test-bucket/job-document.json",
    "job": {
        "status": "IN_PROGRESS",
        "jobArn": "arn:aws:iot:us-east-1:123456789012:job/010",
        "targets": [
            "arn:aws:iot:us-east-1:123456789012:thing/myThing"
        ],
        "jobProcessDetails": {
            "numberOfCanceledThings": 0,
            "numberOfFailedThings": 0,
            "numberOfInProgressThings": 0,
            "numberOfQueuedThings": 0,
            "numberOfRejectedThings": 0,
            "numberOfRemovedThings": 0,
            "numberOfSucceededThings": 0,
            "processingTargets": [
                arn:aws:iot:us-east-1:123456789012:thing/thingOne,
                arn:aws:iot:us-east-1:123456789012:thinggroup/thinggroupOne,
                arn:aws:iot:us-east-1:123456789012:thing/thingTwo,
                arn:aws:iot:us-east-1:123456789012:thinggroup/thinggroupTwo
            ]
        },
        "presignedUrlConfig": {
            "expiresInSec": 60,
            "roleArn": "arn:aws:iam::123456789012:role/S3DownloadRole"
        },
        "jobId": "010",
```

```
        "lastUpdatedAt": 1486593195.006,
        "createdAt": 1486593195.006,
        "targetSelection": "SNAPSHOT"
    }
}
```

List Executions for a Job

ListJobExecutionsForJob

A job running on a specific device is represented by a job execution object. You use the `ListJobExecutionsForJob` command to list all job executions for a job.

more info (8)

```
aws iot list-job-executions-for-job --job-id 010
```

The command returns a list of job executions:

```
{
    "executionSummaries": [
        {
            "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingOne",
            "jobExecutionSummary": {
                "status": "QUEUED",
                "lastUpdatedAt": 1486593196.378,
                "queuedAt": 1486593196.378,
                "executionNumber": 1234567890
            }
        },
        {
            "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingTwo",
            "jobExecutionSummary": {
                "status": "IN_PROGRESS",
                "lastUpdatedAt": 1486593345.659,
                "queuedAt": 1486593196.378,
                "startedAt": 1486593345.659,
                "executionNumber": 4567890123
            }
        }
    ]
}
```

List Job Executions for a Thing

ListJobExecutionsForThing

You use the `ListJobExecutionsForThing` command to list all job executions running on a thing.

more info (9)

```
aws iot list-job-executions-for-thing --thing-name thingOne
```

The command returns a list of job executions that are running or have run on the specified thing:

```
{
    "executionSummaries": [
```

```
        {
            "jobExecutionSummary": {
                "status": "QUEUED",
                "lastUpdatedAt": 1486687082.071,
                "queuedAt": 1486687082.071,
                "executionNumber": 9876543210
            },
            "jobId": "013"
        },
        {
            "jobExecutionSummary": {
                "status": "IN_PROCESS",
                "startAt": 1486685870.729,
                "lastUpdatedAt": 1486685870.729,
                "queuedAt": 1486685870.729,
                "executionNumber": 1357924680
            },
            "jobId": "012"
        },
        {
            "jobExecutionSummary": {
                "status": "COMPLETED",
                "startAt": 1486678853.415,
                "lastUpdatedAt": 1486678853.415,
                "queuedAt": 1486678853.415,
                "executionNumber": 4357680912
            },
            "jobId": "011"
        },
        {
            "jobExecutionSummary": {
                "status": "CANCELED",
                "startAt": 1486593196.378,
                "lastUpdatedAt": 1486593196.378,
                "queuedAt": 1486593196.378,
                "executionNumber": 2143174250
            },
            "jobId": "010"
        }
    ]
}
```

Describe Job Execution

DescribeJobExecution

You use the `DescribeJobExecution` command to get the status of a specific job execution. You specify a job ID and thing name (and, optionally, an execution number) to identify the job execution.

The job's execution status must be `QUEUED` or `IN_PROGRESS`.

more info (10)

```
aws iot describe-job-execution --job-id 017 --thing-name thingOne
```

The command returns the JobExecution (p. 280):

```
{
    "execution": {
        "jobId": "017",
        "executionNumber": 4516820379,
```

```
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingOne",
        "createdAt": 1489084805.285,
        "lastUpdatedAt": 1489086279.937,
        "startedAt": 1489086279.937,
        "status": "IN_PROGRESS",
        "statusDetails": {
            "status": "IN_PROGRESS",
            "detailsMap": {
                "percentComplete": "10"
            }
        }
    }
}
```

Jobs Events

Jobs also publishes to reserved topics on the MQTT protocol when jobs are pending, completed, or canceled, and when a device reports success or failure when executing a job. Devices or management and monitoring applications can keep track of the status of jobs by subscribing to these topics.

For more information about publishing and subscribing to MQTT topics, see Message Broker for AWS IoT (p. 151)

job pending

AWS IoT Jobs publishes a message on an MQTT topic when a job is added to or removed from the list of pending job executions for a thing, or there is a change to the order to the jobs on the list:

- `$aws/things/thingName/jobs/notify`
- `$aws/things/thingName/jobs/notify-next`

more info (11)

The messages contain the following example payloads:

`$aws/things/thingName/jobs/notify`:

```
{
    "jobs" : {
        "JobExecutionState": [ JobExecutionSummary (p. 281) ... ],
    },
    "timestamp": timestamp,
}
```

`$aws/things/thingName/jobs/notify-next`:

```
{
    "execution" : JobExecutionData (p. 290),
    "timestamp": timestamp,
}
```

job completed/canceled

AWS IoT Jobs publishes a message on an MQTT topic when a job is completed or canceled:

- `$aws/events/job/jobID/completed`

- `$aws/events/job/jobID/canceled`

more info (12)

The message contains the following example payload:

```
{
    "eventType": "job",
    "eventId": "UUID",
    "timestamp": timestamp,
    "operation": "completed|canceled",
    "jobId":  "043",
    "status": "COMPLETED|CANCELED",
    "targets": [
        "arn:aws:iot:us-west-2:123456789012:thing/xxxxxx",
        "arn:aws:iot:us-west-2:123456789012:thing/yyyyyy",
        "arn:aws:iot:us-west-2:123456789012:thing/zzzzzz"
    ],
    "description": "sample description",
    "completedAt": "14889914167084",
    "createdAt": "14889025672199",
    "lastUpdatedAt": "14889734904359",
    "jobProcessDetails": {
        "numberOfCanceledThings": 1,
        "numberOfSucceededThings": 1,
        "numberOfRejectedThings": 0,
        "numberOfFailedThings": 1,
        "numberOfInProgressThings": 0,
        "numberOfRemovedThings": 0,
        "processingTargets": [
            arn:aws:iot:us-east-1:123456789012:thing/thingOne,
            arn:aws:iot:us-east-1:123456789012:thing/thingTwo,
            arn:aws:iot:us-east-1:123456789012:thinggroup/thingThree
        ]
    }
}
```

job execution terminal status

AWS IoT Jobs publishes a message when a device updates a job execution to terminal status:

- `$aws/events/jobExecution/jobID/succeeded`
- `$aws/events/jobExecution/jobID/failed`
- `$aws/events/jobExecution/jobID/rejected`
- `$aws/events/jobExecution/jobID/canceled`

more info (13)

The message contains the following example payload:

```
{
    "eventType": "jobExecution",
    "eventId": "UUID",
    "timestamp": "14889025672199",
    "operation": "succeeded|failed|rejected|canceled",
    "jobId":  "031",
    "status": "SUCCESS|FAILED|REJECTED|CANCELED",
    "thingName": "myThing",
```

```
}
```

Devices and Jobs

device communication with Jobs

Devices can communicate with Jobs through the use of MQTT messages. Each device must subscribe to the appropriate topics to receive messages from Jobs. Each device has its own MQTT topic. Devices can also use an HTTPS API to communicate with Jobs. This section describes how to use MQTT messages to program a device to work with Jobs. The HTTPS API commands are similar and are described in Using the AWS IoT Jobs APIs (p. 276).

using the MQTT protocol

Communication between the Jobs service and your devices can occur over the MQTT protocol. Devices subscribe to MQTT topics to be notified of new jobs and receive responses from the Jobs service. Devices publish on MQTT topics to query or update the state of a job execution.

MQTT topic overview

For more information about publishing and subscribing to MQTT topics, see Message Broker for AWS IoT (p. 151)

Devices can:

- Be notified whenever a job execution is added or removed from the list of pending job executions for a device by subscribing to the `$aws/things/thing-name/jobs/notify` MQTT topic, where `thing-name` is the name of the thing associated with the device.
- Be notified when the next pending job execution has changed by subscribing to the `$aws/things/thing-name/jobs/notify-next` MQTT topic, where `thing-name` is the name of the thing associated with the device.
- Update the status of a job execution by calling the UpdateJobExecution (p. 299) API.
- Query the status of a job execution by calling the DescribeJobExecution (p. 297) API.
- Retrieve a list of pending job executions by calling the GetPendingJobExecutions (p. 294) API.
- Retrieve the next pending job execution by calling the DescribeJobExecution (p. 297) API with jobId `$next`.
- Get and start the next pending job execution by calling the StartNextPendingJobExecution (p. 295) API.

The Jobs service publishes success and failure messages on an MQTT topic formed by appending `accepted` or `rejected` to the topic used to make the request. For example, if a request message is published on the `$aws/things/myThing/jobs/get` topic, the Jobs service publishes success messages on the `$aws/things/myThing/jobs/get/accepted` topic and publishes rejected messages on the `$aws/things/myThing/jobs/get/rejected` topic.

more info (14)

In this section, a device called MyThing should subscribe to the following MQTT topics:

- `$aws/things/MyThing/jobs/notify` (or `$aws/things/MyThing/jobs/notify-next`)
- `$aws/things/MyThing/jobs/get/accepted`
- `$aws/things/MyThing/jobs/get/rejected`
- `$aws/things/MyThing/jobs/jobId/get/accepted`
- `$aws/things/MyThing/jobs/jobId/get/rejected`

Programming Devices to Work with Jobs

Device Workflow

In general, there are two ways a device can handle the jobs it is given to execute.

Option A: Get the next job

1. When a device first comes online, it should subscribe to the device's `notify-next` topic.
2. Call the DescribeJobExecution (p. 297) MQTT API with jobId `$next` to get the next job, its job document, and other details, including any state saved in `statusDetails`.
3. Call the UpdateJobExecution (p. 299) MQTT API to update the job status. Or, to combine this and the previous step in one call, the device can call StartNextPendingJobExecution (p. 295).
4. Perform the actions specified by the job document using the UpdateJobExecution (p. 299) MQTT API to report on the progress of the job.
5. Call the UpdateJobExecution (p. 299) MQTT API again to update the job status and report success or failure.
6. Because this job's execution status has been changed to a terminal status, the next job available for execution (if any) will change. The device is notified that the next pending job execution has changed. At this point, the device should continue as described in step 2.

   If the device remains online, it will continue to receive a notifications of the next pending job execution, including its JobExecutionData, whenever it completes a job or when a new pending job execution is added. When this occurs, the device continues as described in step 2.

Option B: Pick from available jobs

1. When a device first comes online, it should subscribe to the thing's `notify` topic.
2. Call the GetPendingJobExecutions (p. 294) MQTT API to get a list of pending job executions.
3. If the list contains one or more job executions, pick one.
4. Call the DescribeJobExecution (p. 297) MQTT API to get the job document and other details, including any state saved in statusDetails.
5. Call the UpdateJobExecution (p. 299) MQTT API to update the job status. If the `includeJobDocument` field is set to true in this command, the device can skip the previous step and retrieve the job document at this point.
6. Perform the actions specified by the job document using the UpdateJobExecution (p. 299) MQTT API to report on the progress of the job.
7. Call the UpdateJobExecution (p. 299) MQTT API again to update the job status and report success or failure.

   If the device remains online, it will be notified of all pending job executions whenever a new pending job execution becomes available. When this occurs, the device can continue as described in step 2.

If the device is unable to execute the job, it should call the UpdateJobExecution (p. 299) MQTT API to update the job status to `REJECTED`.

Starting a New Job

new job notification

When a new job is created, Jobs publishes a message on the `$aws/things/`*`thing-name`*`/jobs/notify` topic for each target device.

more info (15)

The message contains the following information:

```
{
    "timestamp":1476214217017,
    "jobs":{
        "QUEUED":[{
            "jobId":"0001",
            "queuedAt":1476214216981,
            "lastUpdatedAt":1476214216981,
            "versionNumber" : 1
        }]
    }
}
```

The device receives this message on the '$aws/things/*thingName*/jobs/notify' topic when the job execution is queued.

get job information

To get more information about a job execution, the device calls the DescribeJobExecution (p. 297) MQTT API with the `includeJobDocument` field set to true.

more info (16)

If the request is successful, Jobs publishes a message on the `$aws/things/MyThing/jobs/0023/get/accepted` topic:

```
{
    "clientToken" : "client-001",
    "timestamp" : 1489097434407,
    "execution" : {
        "jobId" : "023",
        "status" : "QUEUED",
        "queuedAt" : 1489097374841,
        "lastUpdatedAt" : 1489097374841,
        "versionNumber" : 1,
        "jobDocument" : {
            < contents of job document >
        }
    }
}
```

Note
If the request fails, Jobs publishes a message on the `$aws/things/MyThing/jobs/0023/get/rejected` topic.

The device now has the job document, which it can interpret to perform the remote operations for the job. If the job document contains an Amazon S3 presigned URL, the device can use that URL to download any required files for the job.

The device can call the StartNextPendingJobExecution (p. 295) MQTT API to request more information and start any pending job execution, in one step.

Report Job Execution Status

update execution status

As the device is executing the job, it can call the UpdateJobExecution (p. 299) MQTT API to update the status of the job execution.

more info (17)

For example, a device can update the job execution status to `IN_PROGRESS` by publishing the following message on the `$aws/things/MyThing/jobs/0023/update` topic:

```
{
    "status":"IN_PROGRESS",
    "statusDetails": {
        "progress":"50%"
    },
    "expectedVersion":"1",
    "clientToken":"client001"
}
```

Jobs responds by publishing a message to the `$aws/things/MyThing/jobs/0023/update/accepted` or `$aws/things/MyThing/jobs/0023/update/rejected` topic:

```
{
    "clientToken":"client001",
    "timestamp":1476289222841
}
```

The device can combine the two previous requests by calling StartNextPendingJobExecution (p. 295), which gets and starts the next pending job execution and allows the device to update the job execution status. This request also returns the job document when there is a job execution pending.

The `status` field can be set to `QUEUED`, `IN_PROGRESS`, `SUCCESS`, `FAILED`, `REJECTED`, `REMOVED`, or `CANCELED`. You cannot update the status of a job execution that is already in a terminal state.

report execution completed

When the device is finished executing the job, it calls the UpdateJobExecution (p. 299) MQTT API. If the job was successful, set `status` to `SUCCESS` and, in `statusDetails` in the message payload, add other information about the job as name/value pairs.

more info (18)

For example:

```
{
    "status":"SUCCESS",
    "statusDetails": {
        "progress":"100%"
    },
    "expectedVersion":"2",
    "clientToken":"client-001"
}
```

If the job was not successful, set `status` to `FAILED` and, in `statusDetails`, add information about the error that occurred:

```
{
```

```
    "status":"FAILED",
    "statusDetails": {
        "errorCode":"101",
        "errorMsg":"Unable to install update"
    },
    "expectedVersion":"2",
    "clientToken":"client-001"
}
```

Note
The `statusDetails` attribute can contain any number of name/value pairs.

When Jobs receives this update, it publishes a message on the `$aws/things/MyThing/jobs/notify` topic to indicate the job execution is complete:

```
{
    "timestamp":1476290692776,
    "jobs":{}
}
```

Additional Jobs

additional jobs

If there are other job executions pending for the device, they are included in the message published to `$aws/things/MyThing/jobs/notify`.

more info (19)

For example:

```
{
    "timestamp":1476290692776,
    "jobs":{
        "QUEUED":[{
            "jobId":"0002",
            "queuedAt":1476290646230,
            "lastUpdatedAt":1476290646230
        }],
        "IN_PROCESS":[{
            "jobId":"0003",
            "queuedAt":1476290646230,
            "lastUpdatedAt":1476290646230
        }]
    }
}
```

Using the AWS IoT Jobs APIs

There are two categories of API used in AWS IoT Jobs:

- Those used for management and control of jobs.
- Those used by the devices executing those jobs.

In general, job management and control uses an HTTPS protocol API. Devices can use either an MQTT or an HTTPS protocol API. (The HTTPS API is designed for a low volume of calls typical when creating and

tracking jobs. It usually opens a connection for a single request, and then closes the connection after the response is received. The MQTT API allows long polling. It is designed for large amounts of traffic that can scale to millions of devices.)

Note
Each Jobs HTTPS API has a corresponding command that allows you to call the API from the AWS CLI. The commands are lowercase, with hyphens between the words that make up the name of the API. For example, you can invoke the `CreateJob` API on the CLI by typing:

```
aws iot create-job ...
```

Job Management and Control API

Job Management and Control Data Types

The following data types are used by management and control applications to communicate with Jobs.

Job

Job data type

The `Job` object contains details about a job.

syntax (1)

```
{
    "jobArn": "string",
    "jobId": "string",
    "status": "IN_PROGRESS|CANCELED|COMPLETED",
    "targetSelection": "CONTINUOUS|SNAPSHOT",
    "comment": "string",
    "targets": ["string"],
    "description": "string",
    "createdAt": timestamp,
    "lastUpdatedAt": timestamp,
    "completedAt": timestamp,
    "jobProcessDetails": {
        "processingTargets": ["string"],
        "numberOfCanceledThings": "long",
        "numberOfSucceededThings": "long",
        "numberOfFailedThings": "long",
        "numberOfRejectedThings": "long",
        "numberOfQueuedThings": "long",
        "numberOfInProgressThings": "long",
        "numberOfRemovedThings": "long",
    },
    "presignedUrlConfig": {
        "expiresInSec": number,
        "roleArn": "string"
    },
    "jobExecutionRolloutConfig": {
        "maximumPerMinute": "integer"
    }
}
```

description (1)

`jobArn`

An ARN identifying the job with the format "arn:aws:iot:*region*:*account*:job/*jobId*".

`jobId`

The unique identifier you assigned to this job when it was created.

`status`

The status of the job, one of `IN_PROGRESS`, `CANCELED`, or `COMPLETED`.

`targetSelection`

Specifies whether the job continues to run (CONTINUOUS) or is complete after those things specified as targets have completed the job (SNAPSHOT). If CONTINUOUS, the job might also be run on a thing when a change is detected in a target. For example, a job runs on a thing when the thing is added to a target group, even after the job was completed by all things originally in the group.

`comment`

If the job was updated, describes the reason for the update.

`targets`

A list of AWS IoT things and thing groups to which the job should be sent.

`description`

A short text description of the job.

`createdAt`

The time, in milliseconds since the epoch, when the job was created.

`lastUpdatedAt`

The time, in milliseconds since the epoch, when the job was last updated.

`completedAt`

The time, in milliseconds since the epoch, when the job was completed.

`jobProcessDetails`

Details about the job process:

`processingTargets`

A list of AWS IoT things and thing groups that are currently executing the job.

`numberOfCanceledThings`

The number of AWS IoT things that canceled the job.

`numberOfSucceededThings`

The number of AWS IoT things that successfully completed the job.

`numberOfFailedThings`

The number of AWS IoT things that failed to complete the job.

`numberOfRejectedThings`

The number of AWS IoT things that rejected the job.

`numberOfQueuedThings`

The number of AWS IoT things that are awaiting execution of the job.

`numberOfInProgressThings`

The number of AWS IoT things that are currently executing the job.

`numberOfRemovedThings`

The number of AWS IoT things that are no longer scheduled to execute the job because they have been deleted or removed from the group that was a target of the job.

`presignedUrlConfig`

Configuration information for presigned Amazon S3 URLs.

`expiresInSec`

How long (in seconds) presigned URLs are valid. Valid values are 60 - 3600. The default value is 3600 seconds. Presigned URLs are generated when Jobs receives an MQTT request for the job document.

`roleArn`

The ARN of an IAM role that grants permission to download files from an Amazon S3 bucket. The role must also grant permission for AWS IoT to download the files. For more information about how to create and configure the role, see Job Documents (p. 263).

`jobExecutionRolloutConfig`

Allows you to create a staged rollout of a job.

`maximumPerMinute`

The maximum number of things (devices) to which the job will be sent for execution, per minute.

JobSummary

JobSummary data type

The `JobSummary` object contains a job summary.

syntax (2)

```
{
    "jobArn": "string",
    "jobId": "string",
    "status": "IN_PROGRESS|CANCELED|COMPLETED",
    "description": "string",
    "targetSelection": "CONTINUOUS|SNAPSHOT",
    "thingGroupId": "string",
    "createdAt": timestamp,
    "lastUpdatedAt": timestamp,
    "completedAt": timestamp
}
```

description (2)

`jobArn`

An ARN that identifies the job.

`jobId`

The unique identifier you assigned to this job when it was created.

`status`

The job status. Can be one of `IN_PROGRESS`, `CANCELED`, or `COMPLETED`.

`targetSelection`

Specifies whether the job continues to run (CONTINUOUS) or is complete after all those things specified as targets have completed the job (SNAPSHOT). If CONTINUOUS, the job might also be run on a thing when a change is detected in a target. For example, a job runs on a thing when the thing is added to a target group, even after the job was completed by all things originally in the group.

`thingGroupId`

The ID of the thing group.

`createdAt`

The UNIX timestamp for when the job was created.

`lastUpdatedAt`

The UNIX timestamp for when the job was last updated.

`completedAt`

The UNIX timestamp for when the job was completed.

JobExecution

JobExection data type

The `JobExecution` object represents the execution of a job on a particular device.

syntax (3)

```
{
    "jobId": "string",
    "executionNumber": 1234567890,
    "thingArn": "string",
    "queuedAt": timestamp,
    "lastUpdatedAt": timestamp,
    "startedAt": timestamp,
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCESS|CANCELED|REJECTED|REMOVED",
    "statusDetails": {
        "detailsMap": {
            "string": "string" ...
        },
        "status": "string"
    },
}
```

description (3)

`jobId`

The unique identifier you assigned to this job when it was created.

`executionNumber`

A number that identifies this job execution on this particular device. It can be used later in commands that return or update job execution information.

`thingArn`

The AWS IoT thing ARN.

`queuedAt`

The time, in milliseconds since the epoch, when the job execution was queued.

`lastUpdatedAt`

The time, in milliseconds since the epoch, when the job execution was last updated.

`startedAt`

The time, in milliseconds since the epoch, when the job execution was started.

`status`

The status of the job execution. Can be one of "QUEUED", "IN_PROGRESS", "FAILED", "SUCCESS", "CANCELED", "REJECTED", or "REMOVED".

`statusDetails`

A collection of name/value pairs that describe the status of the job execution.

JobExecutionSummary

JobExecutionSummary data type

The `JobExecutionSummary` object contains job execution summary information:

syntax (4)

```
{
    "executionNumber": 1234567890,
    "queuedAt": timestamp,
    "lastUpdatedAt": timestamp,
    "startedAt": timestamp,
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCESS|CANCELED|REJECTED|REMOVED"
}
```

description (4)

`executionNumber`

A number that identifies a particular job execution on a particular device. It can be used later in commands that return or update job execution information.

`queuedAt`

The time, in milliseconds since the epoch, when the job execution was queued.

`lastUpdatedAt`

The time, in milliseconds since the epoch, when the job execution was last updated.

`startAt`

The time, in milliseconds since the epoch, when the job execution was started.

`status`

The status of the job execution: QUEUED, IN_PROGRESS, FAILED, SUCCESS, CANCELED, REJECTED, or REMOVED.

JobExecutionSummaryForJob

JobExecutionSummaryForJob data type

The `JobExecutionSummaryForJob` object contains a summary of information about job executions for a specific job.

syntax (5)

```
{
    "executionSummary": [
        {
            "thingArn": "string",
            "jobExecutionSummary": { JobExecutionSummary }
        }
        ...
    ]
}
```

description (5)

`thingArn`

The AWS IoT thing ARN.

`jobExecutionSummary`

An JobExecutionSummary (p. 281) object.

JobExecutionSummaryForThing

JobExecutionSummaryForThing data type

The `JobExecutionSummaryForThing` object contains a summary of information about a job execution on a specific thing.

syntax (6)

```
{
    "jobId": "string",
    "jobExecutionSummary": { JobExecutionSummary }
}
```

description (6)

`jobId`

The unique identifier you assigned to this job when it was created.

`jobExecutionSummary`

A JobExecutionSummary (p. 281) object.

Job Management and Control HTTPS Commands

The following commands are available for management and control applications over the HTTPS protocol.

AssociateTargetsWithJob

AssociateTargetsWithJob command

Associates a group with a continuous job. For more information, see CreateJob (p. 284). The following criteria must be met:

- The job must have been created with the `targetSelection` field set to "CONTINUOUS".
- The job status must currently be "IN_PROGRESS".

- The total number of targets associated with a job must not exceed 100.

HTTPS (1)

Request:

```
POST /jobs/jobId/targets

{
    "targets": [ "string" ],
    "comment": "string"
}
```

`jobId`

The unique identifier you assigned to this job when it was created.

`targets`

A list of thing group ARNs that define the targets of the job.

`comment`

Optional. A comment string that describes why the job was associated with the targets.

Response:

```
{
    "jobArn": "string",
    "jobId": "string",
    "description": "string"
}
```

`jobArn`

An ARN identifying the job.

`jobId`

The unique identifier you assigned to this job when it was created.

`description`

A short text description of the job.

MQTT (1)

Not available.

CancelJob

CancelJob command

Cancels a job.

HTTPS (2)

Request:

```
PUT /jobs/jobId/cancel
```

```
{
    "comment": "string"
}
```

`jobId`

The unique identifier you assigned to this job when it was created.

`comment`

[Optional] A comment string describing why the job was canceled.

Response:

```
{
    "jobArn": "string",
    "jobId": "string",
    "description": "string"
}
```

`jobArn`

The job ARN.

`jobId`

The unique identifier you assigned to this job when it was created.

`description`

A short text description of the job.

MQTT (2)

Not available.

CreateJob

CreateJob command

Creates a job. You can provide the job document as a link to a file in an Amazon S3 bucket (`documentSource` parameter) or in the body of the request (`document` parameter).

A job can be made *continuous* by setting the optional `targetSelection` parameter to "CONTINUOUS". (The default is "SNAPSHOT".) A continuous job can be used to onboard or upgrade devices as they are added to a group because it continues to run and is executed on newly added things, even after the things in the group at the time the job was created have completed the job.

The following validations are performed on arguments to the `CreateJob` API:

- The `targets` argument must be a list of valid thing or thing group ARNs. All things and thing groups must be in your AWS account.
- The `documentSource` argument must be a valid Amazon S3 URL to a job document. Amazon S3 URLs are of the form: `https://s3.amazonaws.com/`*`bucketName`*`/`*`objectName`*.
- The document stored in the URL specified by the `documentSource` argument must be a UTF-8 encoded JSON document.
- The size of a job document is limited to 32 KB due to the limit on the size of an MQTT message(128 KB) and encryption.
- The `jobId` must be unique within your AWS account.

HTTPS (3)

Request:

```
PUT /jobs/jobId

{
    "targets": [ "string" ],
    "document": "string",
    "documentSource": "string",
    "description": "string",
    "presignedUrlConfigData": {
        "roleArn": "string",
        "expiresInSec": "integer"
    },
    "targetSelection": "CONTINUOUS|SNAPSHOT",
    "jobExecutionsRolloutConfig": {
        "maximumPerMinute": "integer"
    }
}
```

`jobId`

A job identifier, which must be unique for your AWS account. We recommend using a UUID. Alpha-numeric characters, "-", and "_" can be used here.

`targets`

A list of thing or thing group ARNs that defines the targets of the job.

`document`

Optional. The job document.

`documentSource`

Optional. An Amazon S3 link to the job document.

`description`

Optional. A short text description of the job.

`presignedUrlConfigData`

Optional. Configuration information for presigned Amazon S3 URLs.

`roleArn`

The ARN of the IAM role that contains permissions to access the Amazon S3 bucket. This is the bucket that contains the data that devices download with the presigned Amazon S3 URLs. This role must also grant AWS IoT permission to assume the role. For more information, see Job Documents (p. 263).

`expiresInSec`

How long (in seconds) presigned URLs are valid. Valid values are 60 - 3600. The default value is 3600 seconds. Presigned URLs are generated when Jobs receives an MQTT request for the job document.

`targetSelection`

Optional. Specifies whether the job continues to run (CONTINUOUS) or is complete after all those things specified as targets have completed the job (SNAPSHOT). If CONTINUOUS, the job might also be scheduled to run on a thing when a change is detected in a target. For example, a job is scheduled to run on a thing when the thing is added to a target group, even after the job was completed by all things originally in the group.

`jobExecutionRolloutConfig`

Optional. Allows you to create a staged rollout of a job.

`maximumPerMinute`

The maximum number of things on which the job is sent for execution, per minute. Valid values are 1 to 1000. If not specified, the default is 1000. The actual number of things that receive the job might be less during any particular minute interval (due to system latency), but will not be more than the specified value.

Response:

```
{
    "jobArn": "string",
    "jobId": "string",
    "description": "string"
}
```

`jobArn`

The ARN of the job.

`jobId`

The unique identifier you assigned to this job.

`description`

An optional short text description of the job.

MQTT (3)

Not available.

DescribeJob

DescribeJob command

Gets the details of the specified job.

HTTPS (4)

Request:

```
GET /jobs/jobId
```

`jobId`

The unique identifier you assigned to this job when it was created.

Response:

```
{
    "documentSource": "string",
    "job": Job
}
```

`documentSource`

An Amazon S3 link to the job document.

`job`

A Job (p. 277) object.

MQTT (4)

Not available.

DescribeJobExecution

DescribeJobExecution command

Gets details of a job execution. The job's execution status must be `SUCCEEDED` or `FAILED`.

HTTPS (5)

Request:

```
GET /things/thingName/jobs/jobId?executionNumber=executionNumber
```

`jobId`

The unique identifier you assigned to this job when it was created.

`thingName`

The thing name associated with the device the job execution is running on.

`executionNumber`

Optional. A number that is used to specify a particular job execution on a particular device. (See JobExecution (p. 280).) If not specified, the latest job execution is returned.

Response:

```
{
    "execution": { JobExecution }
}
```

`execution`

A JobExecution (p. 280) object.

MQTT (5)

Not available.

GetJobDocument

GetJobDocument command

Gets the job document for a job.

Note
Placeholder URLs are not replaced with presigned Amazon S3 URLs in the document returned. Presigned URLs are generated only when Jobs receives a request over MQTT.

HTTPS (6)

Request:

```
GET /jobs/jobId/job-document
```

`jobId`

The unique identifier you assigned to this job when it was created.

Response:

```
{
    "document": "string"
}
```

`document`

The job document content.

MQTT (6)

Not available.

ListJobExecutionsForJob

ListExecutionsForJob command

Gets a list of job executions for a job.

HTTPS (7)

Request:

```
GET /jobs/jobId/things?status=status&maxResults=maxResults&nextToken=nextToken
```

`jobId`

The unique identifier you assigned to this job when it was created.

`status`

Optional. A filter that lets you search for jobs that have the specified status: QUEUED, IN_PROGRESS, SUCCESS, FAILED, REJECTED, REMOVED, or CANCELED.

`maxResults`

Optional. The maximum number of results to be returned per request.

`nextToken`

Optional. The token to retrieve the next set of results.

Response:

```
{
    "executionSummaries": [ JobExecutionSummary ... ]
}
```

`executionSummaries`

A list of JobExecutionSummary (p. 281) objects associated with the specified job ID.

MQTT (7)

Not available.

ListJobExecutionsForThing

ListJobExecutionsForThing command

Gets a list of job executions for a thing.

HTTPS (8)

Request:

```
GET /things/thingName/jobs?status=status&maxResults=maxResults&nextToken=nextToken
```

`thingName`

The name of the thing for which `JobExecutions` will be listed.

`status`

An optional filter that lets you search for jobs that have the specified status: QUEUED, IN_PROGRESS, SUCCESS, FAILED, REJECTED, REMOVED, or CANCELED.

`maxResults`

The maximum number of results to be returned per request.

`nextToken`

The token for the next set of results, or `null` if there are no additional results.

Response:

```
{
    "executionSummaries": [ JobExecutionSummary ... ]
}
```

`executionSummaries`

A list of the JobExecutionSummary (p. 281) objects for the job executions associated with the specified thing.

MQTT (8)

Not available.

ListJobs

ListJobs command

Gets a list of the jobs in your AWS account.

HTTPS (9)

Request:

```
GET /jobs?
status=status&targetSelection=targetSelection&thingGroupName=thingGroupName&thingGroupId=thingGroup
```

`status`

Optional. A filter that lets you search for jobs that have the specified status: IN_PROGRESS, CANCELED, or COMPLETED.

`targetSelection`

Optional. A filter that lets you search for jobs that have the specified `targetSelection` value: CONTINUOUS or SNAPSHOT.

`thingGroupName`

Optional. A filter that lets you search for jobs that have the specified thing group name as a target.

`thingGroupId`

Optional. A filter that lets you search for jobs that have the specified thing group ID as a target.

`maxResults`

Optional. The maximum number of results to be returned per request.

`nextToken`

Optional. The token to retrieve the next set of results.

Response:

```
{
    "jobs": [ JobSummary ... ],
}
```

`jobs`

A list of JobSummary (p. 279) objects, one for each job in your AWS account that matches the specified filtering criteria.

MQTT (9)

Not available.

Jobs Device MQTT and HTTPS APIs

Device MQTT and HTTPS Data Types

The following data types are used to communicate with Jobs over the MQTT and HTTPS protocols.

JobExecution

JobExecution data type

Contains data about a job execution.

syntax (7)

```
{
    "jobId" : "string",
    "thingName" : "string",
    "jobDocument" : "string",
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCESS|CANCELED|REJECTED|REMOVED",
    "statusDetails": {
        "string": "string"
    }
    "queuedAt" : "timestamp",
    "startedAt" : "timestamp",
    "lastUpdatedAt" : "timestamp",
    "versionNumber" : "number",
    "executionNumber": "long"
}
```

description (7)

`jobId`

The unique identifier you assigned to this job when it was created.

`thingName`

The name of the thing that is executing the job.

`jobDocument`

The content of the job document.

`status`

The status of the job execution. Can be one of: "QUEUED", "IN_PROGRESS", "FAILED", "SUCCESS", "CANCELED", "REJECTED", or "REMOVED".

`statusDetails`

A collection of name/value pairs that describe the status of the job execution.

`queuedAt`

The time, in milliseconds since the epoch, when the job execution was enqueued.

`startedAt`

The time, in milliseconds since the epoch, when the job execution was started.

`lastUpdatedAt`

The time, in milliseconds since the epoch, when the job execution was last updated.

`versionNumber`

The version of the job execution. Job execution versions are incremented each time they are updated by a device.

`executionNumber`

A number that identifies a particular job execution on a particular device. It can be used later in commands that return or update job execution information.

JobExecutionState

JobExecutionState data type

Contains data about the state of a job execution.

syntax (8)

```
{
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCESS|CANCELED|REJECTED|REMOVED",
    "statusDetails": {
        "string": "string"
        ...
    }
    "versionNumber": "number"
}
```

description (8)

`status`

The status of the job execution. Can be one of: "QUEUED", "IN_PROGRESS", "FAILED", "SUCCESS", "CANCELED", "REJECTED", or "REMOVED".

`statusDetails`

A collection of name/value pairs that describe the status of the job execution.

`versionNumber`

The version of the job execution. Job execution versions are incremented each time they are updated by a device.

JobExecutionSummary

JobExecutionSummary data type

Contains a subset of information about a job execution.

syntax (9)

```
{
    "jobId": "string",
    "queuedAt": timestamp,
    "startedAt": timestamp,
    "lastUpdatedAt": timestamp,
    "versionNumber": "number",
    "executionNumber": "long"
}
```

description (9)

`jobId`

The unique identifier you assigned to this job when it was created.

`queuedAt`

The time, in milliseconds since the epoch, when the job execution was enqueued.

`startedAt`

The time, in milliseconds since the epoch, when the job execution started.

`lastUpdatedAt`

The time, in milliseconds since the epoch, when the job execution was last updated.

`versionNumber`

The version of the job execution. Job execution versions are incremented each time AWS IoT Jobs receives an update from a device.

`executionNumber`

A number that identifies a particular job execution on a particular device.

ErrorResponse

ErrorRepsonse data type

Contains information about an error that occurred during a Jobs operation.

syntax (10)

```
{
    "code": "ErrorCode",
    "message": "string",
    "clientToken": "string",
    "timestamp": timestamp,
    "executionState": JobExecutionState
}
```

description (10)

`code`

ErrorCode can be set to:

InvalidTopic

The request was sent to a topic in the Jobs namespace that does not map to any API.

InvalidJson

The contents of the request could not be interpreted as valid UTF-8-encoded JSON.

InvalidRequest

The contents of the request were invalid. For example, this code is returned when an `UpdateJobExecution` request contains invalid status details. The message contains details about the error.

InvalidStateTransition

An update attempted to change the job execution to a state that is invalid because of the job execution's current state (for example, an attempt to change a request in state SUCCESS to state IN_PROGRESS). In this case, the body of the error message also contains the `executionState` field.

ResourceNotFound

The JobExecution specified by the request topic does not exist.

VersionMismatch

The expected version specified in the request does not match the version of the job execution in Jobs. In this case, the body of the error message also contains the `executionState` field.

InternalError

There was an internal error processing the request.

RequestThrottled

The request was throttled.

TerminalStateReached

Occurs when a command to describe a job is performed on a job that is in a terminal state.

`message`

An error message string.

`clientToken`

An arbitrary string used to correlate a request with its reply.

`timestamp`

The time, in milliseconds since the epoch.

`executionState`

A JobExecutionState (p. 291) object. This field is included only when the `code` field has the value `InvalidStateTransition` or `VersionMismatch`. This makes it unnecessary in these cases to perform a separate `DescribeJobExecution` request to obtain the current job execution status data.

Device Commands

The following commands are available over the MQTT and HTTPS protocols.

GetPendingJobExecutions

GetPendingJobExecutions command

Gets the list of all jobs for a thing that are not in a terminal status.

MQTT (11)

To invoke this API, publish a message on `$aws/things/`*`thingName`*`/jobs/get`.

Request payload:

```
{ "clientToken": "string" }
```

`clientToken`

Optional. A client token used to correlate requests and responses. Enter an arbitrary value here and it will be reflected in the response.

To receive the response, subscribe to:

- `$aws/things/`*`thingName`*`/jobs/get/accepted` ...and...
- `$aws/things/`*`thingName`*`/jobs/get/rejected` ...or...
- `$aws/things/`*`thingName`*`/jobs/get/#` ...for both...

Response payload:

```
{
    "inProgressJobs" : [ JobExecutionSummary ... ],
    "queuedJobs" : [ JobExecutionSummary ... ],
    "timestamp" : 1489096425069,
    "clientToken" : "client-001"
}
```

`inPrograssJobs`

A list of JobExecutionSummary (p. 292) objects with status `IN_PROGRESS`.

`queuedJobs`

A list of JobExecutionSummary (p. 292) objects with status `QUEUED`.

`clientToken`

A client token used to correlate requests and responses.

`timestamp`

The time, in milliseconds since the epoch, when the message was sent.

HTTPS (11)

Request:

```
GET /things/thingName/jobs
```

`thingName`

The name of the thing associated with the device.

Response:

```
{
    "inProgressJobs" : [ JobExecutionSummary ... ],
    "queuedJobs" : [ JobExecutionSummary ... ]
}
```

`inPrograssJobs`

A list of JobExecutionSummary (p. 292) objects.

`queuedJobs`

A list of JobExecutionSummary (p. 292) objects.

StartNextPendingJobExecution

StartNextPendingJobExecution command

Gets and starts the next pending (status IN_PROGRESS or QUEUED) job execution for a thing.

- Any job executions with status IN_PROGRESS are returned first.
- Job executions are returned in the order in which they were created.
- If the next pending job execution is QUEUED, its state is changed to IN_PROGRESS and the job execution's status details are set as specified.
- If the next pending job execution is already IN_PROGRESS, its status details are not changed.
- If no job executions are pending, the response does not include the `execution` field.

MQTT (12)

To invoke this API, publish a message on: `$aws/things/thingName/jobs/start-next`.

Request payload:

```
{
    "statusDetails": {
        "string": "job-execution-state"
        ...
    },
    "clientToken": "string"
}
```

`statusDetails`

A collection of name/value pairs that describe the status of the job execution. If not specified, the `statusDetails` are unchanged.

`clientToken`

A client token used to correlate requests and responses. Enter an arbitrary value here and it will be reflected in the response.

To receive the response, subscribe to:

- `$aws/things/thingName/jobs/start-next/accepted` ...and...
- `$aws/things/thingName/jobs/start-next/rejected` ...or...
- `$aws/things/thingName/jobs/start-next/#` ...for both...

Response payload:

```
{
    "execution" : JobExecutionData,
    "timestamp" : timestamp,
    "clientToken" : "string"
}
```

`execution`

A JobExecution (p. 290) object. For example:

```
{
    "execution" : {
        "jobId" : "022",
        "thingName" : "MyThing",
        "jobDocument" : "< contents of job document >",
        "status" : "IN_PROGRESS",
        "queuedAt" : 1489096123309,
        "lastUpdatedAt" : 1489096123309,
        "versionNumber" : 1,
        "executionNumber" : 1234567890
    },
    "clientToken" : "client-1",
    "timestamp" : 1489088524284,
}
```

`timestamp`

The time, in milliseconds since the epoch, when the message was sent to the device.

`clientToken`

A client token used to correlate requests and responses.

HTTPS (12)

Request:

```
PUT /things/thingName/jobs/$next
{
    "statusDetails": {
        "string": "string"
        ...
    }
}
```

`thingName`

The name of the thing associated with the device.

`statusDetails`

A collection of name/value pairs that describe the status of the job execution. If not specified, the `statusDetails` are unchanged.

Response:

```
{
    "execution" :  JobExecution
}
```

`execution`

A JobExecution (p. 290) object. For example:

```
{
    "execution" : {
        "jobId" : "022",
        "thingName" : "MyThing",
        "jobDocument" : "< contents of job document >",
        "status" : "IN_PROGRESS",
        "queuedAt" : 1489096123309,
        "lastUpdatedAt" : 1489096123309,
        "versionNumber" : 1,
        "executionNumber" : 1234567890
    },
    "clientToken" : "client-1",
    "timestamp" : 1489088524284,
}
```

DescribeJobExecution

DescribeJobExecution command

Gets detailed information about a job execution.

MQTT (13)

To invoke this API, publish a message on `$aws/things/thingName/jobs/jobId/get`.

Request payload:

```
{
```

```
    "executionNumber": "long",
    "includeJobDocument": "boolean",
    "clientToken": "string"
}
```

`thingName`

The name of the thing associated with the device.

`jobId`

The unique identifier assigned to this job when it was created.

Or use `$next` to return the next pending (status IN_PROGRESS or QUEUED) job execution for a thing. In this case, any job executions with status IN_PROGRESS are returned first. Job executions are returned in the order in which they were created.

`executionNumber`

Optional. A number that identifies a particular job execution on a particular device. If not specified, the latest job execution is returned.

`includeJobDocument`

Optional. When set to `true`, the response contains the job document. The default is `true`.

`clientToken`

A client token used to correlate requests and responses. Enter an arbitrary value here and it will be reflected in the response.

To receive the response, subscribe to:

- `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/get/accepted` ...and...
- `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/get/rejected` ...or...
- `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/get/#` ...for both...

Response payload:

```
{
    "execution" : JobExecutionData,
    "timestamp": "timestamp",
    "clientToken": "string"
}
```

`execution`

A JobExecution (p. 290) object.

`timestamp`

The time, in milliseconds since the epoch, when the message was sent.

`clientToken`

A client token used to correlate requests and responses.

HTTPS (13)

The job's execution status must be `QUEUED` or `IN_PROGRESS`.

Request:

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument
```

`thingName`

The name of the thing associated with the device.

`jobId`

The unique identifier assigned to this job when it was created.

Or use `$next` to return the next pending (status IN_PROGRESS or QUEUED) job execution for a thing. In this case, any job executions with status IN_PROGRESS are returned first. Job executions are returned in the order in which they were created.

`includeJobDocument`

Optional. When set to `true`, the response contains the job document. The default is `false`.

`executionNumber`

Optional. A number that identifies a particular job execution on a particular device. If not specified, the latest job execution is returned.

Response:

```
{
    "execution" : JobExecution,
}
```

`execution`

A JobExecution (p. 290) object.

UpdateJobExecution

UpdateJobExecution command

Updates the status of a job execution.

MQTT (14)

To invoke this API, publish a message on `$aws/things/thingName/jobs/jobId/update`.

Request payload:

```
{
    "status": "job-execution-state",
    "statusDetails": {
        "string": "string"
        ...
    },
    "expectedVersion": "number",
    "executionNumber": "long",
    "includeJobExecutionState": "boolean",
    "includeJobDocument": "boolean",
    "clientToken": "string"
}
```

`status`

The new status for the job execution (IN_PROGRESS, FAILED, SUCCEEDED, or REJECTED). This must be specified on every update.

`statusDetails`

A collection of name/value pairs that describe the status of the job execution. If not specified, the `statusDetails` are unchanged.

`expectedVersion`

The expected current version of the job execution. Each time you update the job execution, its version is incremented. If the version of the job execution stored in Jobs does not match, the update is rejected with a `VersionMismatch` error, and an ErrorResponse (p. 293) that contains the current job execution status data is returned. (This makes it unnecessary to perform a separate `DescribeJobExecution` request in order to obtain the job execution status data.)

`executionNumber`

Optional. A number that identifies a particular job execution on a particular device. If not specified, the latest job execution is used.

`includeJobExecutionState`

Optional. When included and set to `true`, the response contains the `JobExecutionState` field. The default is `false`.

`includeJobDocument`

Optional. When included and set to `true`, the response contains the `JobDocument`. The default is `false`.

`clientToken`

A client token used to correlate requests and responses. Enter an arbitrary value here and it will be reflected in the response.

To receive the response, subscribe to:

- `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/update/accepted` ...and...
- `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/update/rejected` ...or...
- `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/update/#` ...for both...

Response payload:

```
{
    "executionState": JobExecutionState,
    "jobDocument": "string",
    "timestamp": timestamp,
    "clientToken": "string"
}
```

`executionState`

A JobExecutionState (p. 291) object.

`jobDocument`

A Job Documents (p. 263) object.

`timestamp`

The time, in milliseconds since the epoch, when the message was sent.

`clientToken`

A client token used to correlate requests and responses.

HTTPS (14)

Request:

```
POST /things/thingName/jobs/jobId
{
    "status": "job-execution-state",
    "statusDetails": {
        "string": "string"
        ...
    },
    "expectedVersion": "number",
    "includeJobExecutionState": "boolean",
    "includeJobDocument": "boolean",
    "executionNumber": "long"
}
```

`thingName`

The name of the thing associated with the device.

`jobId`

The unique identifier assigned to this job when it was created.

`status`

The new status for the job execution (IN_PROGRESS, FAILED, SUCCEEDED, or REJECTED). This must be specified on every update.

`statusDetails`

Optional. A collection of name/value pairs that describe the status of the job execution. If not specified, the `statusDetails` are unchanged.

`expectedVersion`

Optional. The expected current version of the job execution. Each time you update the job execution, its version is incremented. If the version of the job execution stored in Jobs does not match, the update is rejected with a `VersionMismatch` error, and an ErrorResponse (p. 293) that contains the current job execution status data is returned. (This makes it unnecessary to perform a separate `DescribeJobExecution` request in order to obtain the job execution status data.)

`includeJobExecutionState`

Optional. When included and set to `true`, the response contains the JobExecutionState data. The default is `false`.

`includeJobDocument`

Optional. When set to `true`, the response contains the job document. The default is `false`.

`executionNumber`

Optional. A number that identifies a particular job execution on a particular device.

Response:

```
{
    "executionState": JobExecutionState,
    "jobDocument": "string"
}
```

`executionState`

A JobExecutionState (p. 291) object.

`jobDocument`

The contents of the Job Documents (p. 263).

JobExecutionsChanged

JobExecutionsChanged message

Sent whenever a job execution is added to or removed from the list of pending job executions for a thing.

MQTT (15)

Topic: `$aws/things/`*`thingName`*`/jobs/notify`

Message payload:

```
{
    "jobs" : {
        "JobExecutionState": [ JobExecutionSummary (p. 281) ... ]
    },
    "timestamp": timestamp,
}
```

HTTPS (15)

Not available.

NextJobExecutionChanged

NextJobExecutionChanged message

Sent whenever there is a change to which job execution is next on the list of pending job executions for a thing, as defined for DescribeJobExecution (p. 297) with jobId `$next`. This message is not sent when the next job's execution details change, only when the next job that would be returned by `DescribeJobExecution` with jobId `$next` has changed. Consider job executions J1 and J2 with state QUEUED. J1 is next on the list of pending job executions. If the state of J2 is changed to IN_PROGRESS while the state of J1 remains unchanged, then this notification is sent and contains details of J2.

MQTT (16)

Topic: `$aws/things/`*`thingName`*`/jobs/notify-next`

Message payload:

```
{
    "execution" : JobExecutionData (p. 290),
    "timestamp": timestamp,
```

```
}
```

HTTPS (16)

Not available.

Jobs Limits

Resource	Min	Max	Note
`JobId`	1 character	64 characters	The `JobId` length must not exceed 64 characters.
`Document`	N/A	32768 bytes	The maximum size of a document that can be sent to an AWS IoT device is 32 KB.
`DocumentSource`	N/A	1350 characters	The maximum job document source size is 1350 characters.
`Description`	N/A	2028 characters	The maximum job description size is 2028 characters.
`Targets`	1	100	The number of targets a job can have.
`ExpiresInSec`	60 seconds	3600 seconds	The lifetime of pre-signed URLs must be configured greater than 60 seconds and less than 1 hour.
`Comment`	N/A	2028 characters	The maximum comment size is 2028 characters.
`MaxResults`	1	250	The maximum list result per page is 250.
`MaximumJobExecutionsPerMinute`	1	1000	Configures the rollout speed for a job.
Active snapshot jobs	0	100	The maximum number of active snapshot jobs is 100 (irrespective of the number of active continuous jobs).
Active continuous jobs	0	100	The maximum number of active continuous jobs is 100 (irrespective of the number of active snapshot jobs).

Resource	Min	Max	Note
Job document variable substitution	0	10	Up to 10 variables substitutions, including the presign URL, are allowed in a job document.
Data retention	N/A	90 days	Job data and job execution data will be purged after 90 days.
`StatusDetail` map key size	1 character	128 characters	
`StatusDetail` map value size	1 character	128 characters	

Device Provisioning

To provision a device, create a template that describes the resources required for your device. Devices require a thing, a certificate, and one or more policies. A thing is an entry in the device registry that contains attributes that describe a device. Devices use certificates to authenticate with AWS IoT. Policies determine which operations a device can perform in AWS IoT.

Templates contain variables that are replaced when the template is used to provision a device. A dictionary (map) is used to provide values for the variables used in a template. You can use the same template to provision multiple devices. You simply pass in different values for the template variables in the dictionary.

Provisioning Templates

A provision template is a JSON document that uses parameters to describe the resources your device must use to interact with AWS IoT. A template contains two sections: `Parameters` and `Resources`.

Parameters Section

The `Parameters` section declares the parameters used within the `Resources` section. Each parameter declares a name, a type, and an optional default value. The default value is used when the dictionary passed in with the template does not contain a value for the parameter. The `Parameters` section of a template document looks like the following:

```
"Parameters" : {
     "ThingName" : {
       "Type" : "String",
     },
     "SerialNumber" : {
       "Type" : "String",
     },
     "Location" : {
       "Type" : "String",
       "Default" : "WA"
     },
     "CSR" : {
       "Type" : "String",
     }
  }
```

This template snippet declares four parameters: `ThingName`, `SerialNumber`, `Location`, and `CSR`. All of these parameters are of type `String`. The `Location` parameter declares a default value of `"WA"`.

Resources Section

The `Resources` section of the template declares the resources required for your device to communicate with AWS IoT: a thing, a certificate, and one or more policies. Each resource specifies a logical name, a type, and a set of properties.

A logical name allows you to refer to a resource elsewhere in the template.

The type specifies the kind of resource you are declaring. Valid types are:

- `AWS::IoT::Thing`
- `AWS::IoT::Certificate`
- `AWS::IoT::Policy`

The properties you specify depend on the type of resource you are declaring.

Thing Resources

Thing resources are declared using the following properties:

- `ThingName`: String.
- `AttributePayload`: A list of name/value pairs.
- `ThingType`: Optional. String for an associated thing type for the thing.
- `ThingGroups`: Optional. A list of groups to which the thing belongs.

Certificate Resources

Certificates can be specified in one of the following ways:

- A certificate signing request (CSR).
- A certificate ID of an existing device certificate.
- A device certificate created with a CA certificate registered with AWS IoT. If you have more than one CA certificate registered with the same subject field, you must also pass in the CA certificate used to sign the device certificate.

> **Note**
> When you declare a certificate in a template, use only one of the preceding methods. If you use a CSR, you cannot also specify a certificate ID or a device certificate.

For more information about certificates and AWS IoT see AWS IoT and Certificates.

Certificate resources are declared using the following properties:

- `CertificateSigningRequest`: String.
- `CertificatePem`: String.
- `CACertificatePem`: String.
- `Status`: Optional. String that can be one of: ACTIVE, INACTIVE, PENDING_ACTIVATION. Defaults to ACTIVE.

Policy Resources

Policy resources are declared using one the following properties:

- `PolicyName`: Optional. String. Defaults to a hash of the policy document.
- `PolicyDocument`: a stringified JSON object. Optional. If `PolicyDocument` is not provided, the policy must already be created.

Note
If a `Policy` section is present, `PolicyName` or `PolicyDocument`, but not both, must be specified.

Override Settings

If a template specifies a resource that already exists, the `OverrideSettings` section allows you to specify the action to take:

`DO_NOTHING`

Leave the resource as is.

`REPLACE`

Replace the resource with the resource specified in the template.

`FAIL`

Cause the request to fail with a `ResourceConflictsException`.

Resource Example

The following template snippet declares a thing, a certificate, and a policy:

```
"Resources" : {
    "thing" : {
      "Type" : "AWS::IoT::Thing",
      "Properties" : {
        "ThingName" : {"Ref" : "ThingName"},
        "AttributePayload" : { "version" : "v1", "serialNumber" :  {"Ref" :
 "SerialNumber"}},
        "ThingTypeName" :  "lightBulb-versionA",
        "ThingGroups" : ["v1-lightbulbs", {"Ref" : "Location"}]
      }
    },

    "certificate" : {
      "Type" : "AWS::IoT::Certificate",
      "Properties" : {
        "CertificateSigningRequest": {"Ref" : "CSR"},
        "Status" : "ACTIVE"
      }
    },

    "policy" : {
      "Type" : "AWS::IoT::Policy",
      "Properties" : {
        "PolicyDocument" : "{ \"Version\": \"2012-10-17\", \"Statement\": [{ \"Effect
\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\": [\"arn:aws:iot:us-
east-1:123456789012:topic/foo/bar\"] }] }"
      }
    }
  }
```

The thing is declared with:

- The logical name `"thing"`.
- The type `AWS::IoT::Thing`.
- A set of thing properties.

The thing properties include the thing name, a set of attributes, an optional thing type name, and an optional list of thing groups to which the thing will belong.

Parameters are referenced by `{"Ref": "<parameter-name>"}`. When the template is evaluated, the parameters are replaced with the parameter's value from the dictionary passed in with the template.

The certificate is declared with:

- The logical name `"certificate"`.
- The type `AWS::IoT::Certificate`.
- A set of properties.

  The properties include the CSR for the certificate, and setting the status to `ACTIVE`. The CSR text is passed as a parameter in the dictionary passed with the template.

The policy is declared with:

- The logical name `"policy"`.
- The type `AWS::IoT::Policy`.
- Either the name of an existing policy or a policy document.

Template Example

The following JSON file is an example of a complete provisioning template:

(Note that the `PolicyDocument` field value must be a stringified JSON object.)

```
{
  "Parameters" : {
     "ThingName" : {
       "Type" : "String"
     },
     "SerialNumber" : {
       "Type" : "String"
     },
     "Location" : {
       "Type" : "String",
       "Default" : "WA"
     },
     "CSR" : {
       "Type" : "String"
     }
  },

  "Resources" : {

    "thing" : {
      "Type" : "AWS::IoT::Thing",
      "Properties" : {
        "ThingName" : {"Ref" : "ThingName"},
        "AttributePayload" : { "version" : "v1", "serialNumber" :  {"Ref" :
 "SerialNumber"}},
        "ThingTypeName" :  "lightBulb-versionA",
        "ThingGroups" : ["v1-lightbulbs", {"Ref" : "Location"}]
      }
    },
```

```
    "certificate" : {
      "Type" : "AWS::IoT::Certificate",
      "Properties" : {
        "CertificateSigningRequest": {"Ref" : "CSR"},
        "Status" : "ACTIVE"
      }
    },

    "policy" : {
      "Type" : "AWS::IoT::Policy",
      "Properties" : {
        "PolicyDocument" : "{ \"Version\": \"2012-10-17\", \"Statement\": [{ \"Effect
\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\": [\"arn:aws:iot:us-
east-1:123456789012:topic/foo/bar\"] }] }"
      }
    }
  }

}
```

Programmatic Provisioning

To provision a thing, use the RegisterThing API or the `register-thing` CLI command. The `register-thing` CLI command takes the following arguments:

--template-body

The provisioning template.

--parameters

A list of name/value pairs for the parameters used in the provisioning template, in JSON format. For example:`{"ThingName" : "MyProvisionedThing", "CSR" : "<csr-text>"}`

See Template Example (p. 308).

RegisterThing or `register-thing` returns the ARNs for the resources and the text of the certificate it created:

```
{
    "certificatePem": "<certificate-text>",
    "resourceArns": {
        "PolicyLogicalName": "arn:aws:iot:us-
west-2:123456789012:policy/2A6577675B7CD1823E271C7AAD8184F44630FFD7",
        "certificate": "arn:aws:iot:us-west-2:123456789012:cert/
cd82bb924d4c6ccbb14986dcb4f40f30d892cc6b3ce7ad5008ed6542eea2b049",
        "thing": "arn:aws:iot:us-west-2:123456789012:thing/MyProvisionedThing"
    }
}
```

If a parameter is omitted from the dictionary, the default value is used. If no default value is specified, the parameter is not replaced with a value.

Just-in-Time Provisioning

You can have your devices provisioned when they first attempt to connect to AWS IoT. Just-in-time provisioning (JITP) settings are made on CA certificates. You must enable automatic registration and

associate a provisioning template with the CA certificate used to sign the device certificate you are using to provision the device.

You can make these settings when registering a new CA certificate with the RegisterCACertificate API or the `register-ca-certificate` CLI command:

```
aws iot register-ca-certificate --ca-certificate <your-ca-cert> --verification-cert
            <your-verification-cert> --set-as-active --allow-auto-registration --
registration-config <your-template>
```

For more information, see Registering a CA certificate.

You can also use the UpdateCACertificate API or the `update-ca-certificate` CLI command to update the settings for a CA certificate:

```
$ aws iot update-ca-certificate --cert-id <caCertificateId> --new-auto-registration-status
 ENABLE --registration-config <your-template>
```

When a device attempts to connect to AWS IoT by using a certificate signed by a registered CA certificate, AWS IoT loads the template from the CA certificate and calls RegisterThing using the template.

AWS IoT defines the following parameters that you can declare and reference within provisioning templates:

- `AWS::IoT::Certificate::Country`
- `AWS::IoT::Certificate::Organization`
- `AWS::IoT::Certificate::OrganizationalUnit`
- `AWS::IoT::Certificate::DistinguishedNameQualifier`
- `AWS::IoT::Certificate::StateName`
- `AWS::IoT::Certificate::CommonName`
- `AWS::IoT::Certificate::SerialNumber`
- `AWS::IoT::Certificate::Id`

The values for these provisioning template parameters are extracted from the certificate of the device being provisioned.

Bulk Provisioning

You can use the `start-thing-registration-task` command to provision things in bulk. This command takes a provisioning template, an Amazon S3 bucket name, a key name, and a role ARN that allows access to the file in the Amazon S3 bucket. The file in the Amazon S3 bucket contains the values used to replace the parameters in the template. The file must be a newline-delimited JSON file. Each line contains all of the parameter values for provisioning a single device. For example:

```
{"ThingName": "foo", "SerialNumber": "123", "CSR": "csr1"}
{"ThingName": "bar", "SerialNumber": "456", "CSR": "csr2"}
```

Use the ListThingRegistrationTasks API to list the your current bulk thing provisioning tasks. To get information about a specific bulk thing provisioning task, use the DescribeThingRegistrationTask. Use the StopThingRegistrationTask to stop a bulk thing provisioning task.

Note
Only one bulk provisioning operation task can run at a time.

Fleet Indexing Service

Fleet Indexing is a managed service that allows you to index and search your thing registry and thing shadow data in the cloud. After your fleet index is set up, the service manages the indexing of all your thing registry and shadow updates. You can use a simple query language based on the popular open source search engine, Apache Lucene, to search across this data.

To get started, enable indexing and AWS IoT creates the index for your things. After it is active, you can run queries on your index. AWS IoT keeps it continuously updated with your latest data.

You can use the AWS IoT console to manage indexing configuration and run your search queries. If you prefer programmatic access, you can use the AWS SDKs or the AWS CLI.

Please note that there are additional costs for using this service, beyond the standard charges for AWS IoT services, which are outlined in AWS IoT Device Management Pricing.

Managing Indexing

`AWS_Things` is the index created for all of your things. You can control whether you want to index only thing registry data or both thing registry and shadow data.

Enabling Indexing

You can create the `AWS_Things` index and control its configuration by using the `thing-indexing-configuration` setting in the UpdateIndexingConfiguration API. You can retrieve the current indexing configuration by using the GetIndexingConfiguration API.

The following command shows how to use the `get-indexing-configuration` CLI command to retrieve the current thing-indexing configuration:

```
aws iot get-indexing-configuration
{
    "thingIndexing": "OFF"
}
```

The following command shows how to use the AWS IoT `update-indexing-configuration` CLI command to update the thing-indexing configuration:

```
aws iot update-indexing-configuration --thing-indexing-configuration
 thingIndexingMode=REGISTRY_AND_SHADOW
```

Valid values for `thing-indexing-configuration` are:

OFF

No indexing/delete index.

REGISTRY

Create or configure the `AWS_Things` index to index thing registry data only.

REGISTRY_AND_SHADOW

Create or configure the `AWS_Things` index to index thing registry and shadow data.

Note
Normally, thing shadows are encrypted. However, if you decide to include thing shadows in the `AWS_Things` index, the data will be decrypted in order to index it.

Describing Indexes

The following command shows you how to use the `describe-index` CLI command to retrieve the current status of the index:

```
aws iot describe-index --index-name "AWS_Things"
{
    "indexName": "AWS_Things",
    "indexStatus": "BUILDING",
    "schema": "REGISTRY_AND_SHADOW"
}
```

The first time you enable indexing, AWS IoT builds your index. You cannot query the index if the indexStatus is `BUILDING`. The schema indicates which type of data, `REGISTRY` or `REGISTRY_AND_SHADOW`, is indexed.

Changing the configuration for your index causes the index to be rebuilt. The indexStatus during this process is `REBUILDING`. You can execute queries on the existing data while a rebuild is in progress. For example, if you change the index configuration from `REGISTRY` to `REGISTRY_AND_SHADOW`, while the index is being rebuilt, you can query registry data, including the latest updates, but you cannot query the shadow data until the rebuild is complete. The amount of time it takes to build or rebuild the index depends on the amount of data.

Querying an Index

The following command shows how to use the `search-index` CLI command to query data in the index:

```
aws iot search-index --index-name "AWS_Things" --query-string "thingName:mything*"
{
    "things":[{
         "thingName":"mything1",
         "thingGroupNames":[
            "mygroup1"
         ],
         "thingId":"a4b9f759-b0f2-4857-8a4b-967745ed9f4e",
         "attributes":{
            "attribute1":"abc"
         }
    },
    {
        "thingName":"mything2",
        "thingTypeName":"MyThingType",
        "thingGroupNames":[
            "mygroup1",
            "mygroup2"
        ],
        "thingId":"01014ef9-e97e-44c6-985a-d0b06924f2af",
        "attributes":{
            "model":"1.2",
            "country":"usa"
        },
        "shadow":{
            "desired":{
                "location":"new york",
                "myvalues":[3, 4, 5]
            },
```

```
            "reported":{
                "location":"new york",
                "myvalues":[1, 2, 3],
                "stats":{
                    "battery":78
                }
            },
            "metadata":{
                 "desired":{
                       "location":{
                            "timestamp":123456789
                        },
                       "myvalues":{
                             "timestamp":123456789
                       }
                  },
                  "reported":{
                        "location":{
                             "timestamp":34535454
                         },
                        "myvalues":{
                             "timestamp":34535454
                        },
                        "stats":{
                             "battery":{
                                   "timestamp":34535454
                             }
                        }
                  }
            },
            "version":10,
            "timestamp":34535454
        }
    }],
    "nextToken":"AQFCuvk7zZ3D9pOYMbFCeHbdZ+h=G"
}
```


Query Syntax

Queries are specified using a Lucene-like query syntax. For more information, see Lucene query syntax overview on the Apache website.

The following features of Lucene query syntax are supported:

- Terms and phrases
- Searching fields
- Prefix search
- Range search
- Boolean operators AND, OR, NOT and –
- Grouping
- Field grouping
- Escaping special characters

The following features of Lucene query syntax are NOT supported:

- Leading wildcard search (such as "*xyz"), but searching for "*" will match all things
- Regular expressions
- Boosting

- Ranking
- Fuzzy searches
- Proximity search
- Sorting
- Aggregation

A few things to note about the query language:

- The default operator is AND. A query for “thingName:abc thingType:xyz” is equivalent to “thingName:abc AND thingType:xyz”.
- If a field is not specified, AWS IoT searches for the term in all fields.
- All field names are case-sensitive.
- Search is case-insensitive. Words are separated by whitespace characters as defined by Java's `Character.isWhitespace(int)`.
- Indexing of thing shadow data includes reported, desired, delta, and metadata sections.
- Shadow and registry versions are not searchable, but are present in the response.
- The maximum number of terms in a query is 5.

Example Queries

Queries are specified in a query string using a Lucene-like query syntax and passed to the `SearchIndex` API. The following table lists some example query strings:

Query String	Result
"abc"	Queries for "abc" in any registry or thing shadow field.
"thingName:myThingName"	Queries for a thing with name "myThingName".
"thingName:my*"	Queries for things with names that begin with "my".
"thingName:ab?"	Queries for things with names that have "ab" plus one additional character, for example: "aba", "abb", "abc" etc.
"attributes.myAttribute:75"	Queries for things with an attribute called "MyAttribute" that has the value 75.
"attributes.myAttribute:[75 TO 80]"	Queries for things with an attribute called "MyAttribute" whose value falls within a numeric range (75 - 80, inclusive).
"attributes.myAttribute:{75 TO 80]"	Queries for things with an attribute called "MyAttribute" whose value falls within the numeric range (>75 and <=80).
'attributes.serialNumber["abcd" TO "abcf"]'	Queries for things with an attribute called "serialNumber" whose value is within an alphanumeric string range. This query will return things with a "serialNumber" attribute with values "abcd", "abce", or "abcf".

Query String	Result
"attributes.myAttribute:i*t"	Queries for things with an attribute called "MyAttribute" whose value is 'i', followed by any number of characters, followed by 't'.
"attributes.attr1:abc AND attributes.attr2<5 NOT attributes.attr3>10	Queries for things that combine terms using Boolean expressions. This query will return things that have an attribute named "attr1" with a value "abc", an attribute named "attr2" that is less than 5, and an attribute named "attr3" that is not greater than 10.
"shadow.hasDelta:true"	Queries for things whose shadow has a delta element.
"-attributes.model:legacy"	Queries for things where the attribute model is not "legacy".
"shadow.reported.stats.battery:(>70 AND <100) (v2 | v3) -attributes.model:legacy"	Queries for things with the following: • The thing's shadow `stats.battery` attribute has a value between 70 and 100. • The text "v2" or "v3" occurs in a thing's name, type name, or attribute values. • The thing's `model` attribute is not set to "legacy".
"shadow.reported.myvalues:2"	Queries for things where the `myvalues` array in the thing shadow's reported section contains a value of 2.
"shadow.reported.location:* NOT shadow.desired.stats.battery:*"	Queries for things with the following: • The `location` attribute exists in the thing shadow's `reported` section. • The `stats.battery` attribute does not exist in the thing shadow's `desired` section.

Authorization

You can specify the things index as a resource ARN in an AWS IoT policy action:

Action	Resource
iot:SearchIndex	An index ARN (for example, arn:aws:iot:*<your-aws-region>*:index/AWS_Things).
iot:DescribeIndex	An index ARN (for example, arn:aws:iot:*<your-aws-region>*:index/AWS_Things).

AWS IoT Events

AWS IoT publishes event messages when certain events occur. Events are generated by the AWS IoT thing registry when things are added, updated, or deleted. Each event causes a single event message to be sent. Event messages are published over MQTT with a JSON payload. The content of the payload depends on the type of event.

You control which event types are published by calling the UpdateEventConfigurations API.

You can get the current event configuration by calling the DescribeEventConfigurations API.

In order to receive event messages your device must use an appropriate policy that allows it to connect to the AWS IoT device gateway and subscribe to MQTT event topics. You must also subscribe to the appropriate topic filters.

Policy Required for Receiving AWS IoT Events

The following is an example of the policy required for receiving lifecycle events:

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":[
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource":[
            "arn:aws:iot:region:account:/$aws/events/*"
        ]
    }]
}
```

Registry Events

The AWS IoT device registry publishes event messages when things, thing types, and thing groups are created, updated, or deleted. The AWS IoT registry currently supports the following event types:

`thingEvent`

Sent when a thing is created, updated, or deleted.

`thingTypeEvent`

Sent when a thing type is created, deprecated, undeprecated, or deleted.

`thingTypeAssociationMessage`

Sent when a thing type is associated or disassociated with a thing.

`thingGroupEvent`

Sent when a thing group is created, updated, or deleted.

`thingGroupMembershipEvent`

Sent when a thing is added to or removed from a thing group.

`thingGroupHeirarchyEvent`

Sent when adding or removing a thing group to or from its thing group parent.

Registry event messages are published using the following MQTT topics:

`$aws/events/thing/<thingName>/[ created | updated | deleted ]`

Topics where `thing` events are published.

`$aws/events/thingGroup/<groupName>/[ created | updated | deleted ]`

Topics where `thingGroup` events are published.

`$aws/events/thingType/<thingTypeName>/[ created | updated | deleted ]`

Topics where `thingType` events are published.

`$aws/events/thingTypeAssociation/thing/<thingName>/<typeName>`

Topic where `thingTypeAssociation` events are published.

`$aws/events/thingGroupMembership/thingGroup/<thingGroupName>/thing/<thingName>/[ added | removed ]`

Topics where `thingGroupMembership` events are published.

`$aws/events/thingGroupHierarchy/thingGroup/<parentThingGroupName>/childThingGroup/<childThingGroupName>[ added | removed ]`

Topics where `thingGroupHierarchyEvent` events are published.

Note
Event messages are guaranteed to be published once. It is possible for them to be published more than once. The ordering of event messages is not guaranteed.

Thing Events

The payload for a `thingEvent` contains the following attributes:

eventType

Set to "thingEvent" for `thingEvent` events.

eventId

A unique event ID (string).

timestamp

The UNIX timestamp of when the event occurred.

operation

The operation that triggered the event. Valid values are:

- CREATED
- UPDATED
- DELETED

accountId

Your AWS account ID.

thingId

The ID of the thing being created, updated, or deleted.

thingName

The name of the thing being created, updated, or deleted.

versionNumber

The version of the thing being created, updated, or deleted. This value is set to 1 when a thing is created. It is incremented by 1 each time the thing is updated.

thingTypeName

The thing type associated with the thing, if one exists. Otherwise, `null`.

attributes

A collection of name-value pairs associated with the thing.

The following is an example of a payload that is published when a thing has been created.

```
{
    "eventType" : "thingEvent",
    "eventId" : "f5ae9b94-8b8e-4d8e-8c8f-b3266dd89853",
    "timestamp" : 1502210967087,
    "operation" : "CREATED",
    "accountId" : "123456789012",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "thingName" : "MyThing",
    "versionNumber" : 1,
    "thingTypeName" : null,
    "attributes": {
                "attribute3": "value3",
                "attribute1": "value1",
                "attribute2": "value2"
    }
}
```

The following is an example of a payload that is published when a thing has been updated.

```
{
    "eventType" : "thingEvent",
    "eventId" : "b930023e-2348-4f9a-8b75-ba519742bd8a",
    "timestamp" : 1502211188141,
    "operation" : "UPDATED",
    "accountId" : "123456789012",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "thingName" : "MyThing",
    "versionNumber" : 2,
    "thingTypeName" : null,
    "attributes": {
        "attribute3": "value3",
        "attribute1": "value1",
        "attribute2": "value2"
    }
}
```

The following is an example of a payload that is published when a thing has been deleted.

```
{
    "eventType" : "thingEvent",
    "eventId" : "46b0b159-cd50-4b6d-afe7-8b737d997471",
```

```
    "timestamp" : 1502210700975,
    "operation" : "DELETED",
    "accountId" : "123456789012",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "thingName" : "MyThing",
    "versionNumber" : 2,
    "thingTypeName" : null,
    "attributes": {
        "attribute3": "value3",
        "attribute1": "value1",
        "attribute2": "value2"
    }
}
```

Thing Type Events

The payload for a `thingTypeEvent` event contains the following attributes:

eventType

Set to "thingTypeEvent" for `thingTypeEvent` events.

eventId

A unique event ID (string).

timestamp

The UNIX timestamp of when the event occurred.

operation

The operation that triggered the event. Valid values are:

- CREATED
- DEPRECATED
- DELETED

accountId

Your AWS account ID.

thingTypeId

The ID of the thing type being created, deprecated, or deleted.

thingTypeName

The name of the thing type being created, deprecated, or deleted.

isDeprecated

`true` if the thing type is deprecated. Otherwise, `false`.

deprecationDate

The UNIX timestamp for when the thing type was deprecated.

searchableAttributes

A collection of name-value pairs associated with the thing type that can be used for searching.

description

A description of the thing type.

The following is an example of a payload that is published when a thing type is created.

```
{
    "eventType" : "thingTypeEvent",
    "eventId" : "8827376c-4b05-49a3-9b3b-733729df7ed5",
    "timestamp" : 1502140154962,
    "operation" : "CREATED",
    "accountId" : "123456789012",
    "thingTypeId" : "c530ae83-32aa-4592-94d3-da29879d1aac",
    "thingTypeName" : "MyThingType",
    "isDeprecated" : false,
    "deprecationDate" : null,
    "searchableAttributes" : [ "attribute1", "attribute2", "attribute3" ],
    "description" : "My thing type"
}
```

The following is an example of a payload that is published when a thing type is deprecated.

```
{
    "eventType" : "thingTypeEvent",
    "eventId" : "8827376c-4b05-49a3-9b3b-733729df7ed5",
    "timestamp" : 1502140154962,
    "operation" : "UPDATED",
    "accountId" : "123456789012",
    "thingTypeId" : "c530ae83-32aa-4592-94d3-da29879d1aac",
    "thingTypeName" : "MyThingType",
    "isDeprecated" : true,
    "deprecationDate" : 1502212522970,
    "searchableAttributes" : [ "attribute1", "attribute2", "attribute3" ],
    "description" : "My thing type"
}
```

The following is an example of a payload that is published when a thing type is deleted.

```
{
    "eventType" : "thingTypeEvent",
    "eventId" : "8827376c-4b05-49a3-9b3b-733729df7ed5",
    "timestamp" : 1502216007699,
    "operation" : "DELETED",
    "accountId" : "123456789012",
    "thingTypeId" : "c530ae83-32aa-4592-94d3-da29879d1aac",
    "thingTypeName" : "MyThingType",
    "isDeprecated" : true,
    "deprecationDate" : 1502212522970,
    "searchableAttributes" : [ "attribute1", "attribute2", "attribute3" ],
    "description" : "My thing type"
}
```

Thing Group Events

The payload for a `thingGroupEvent` event contains the following attributes:

eventType

Set to "thingGroupEvent" for `thingGroupEvent` events.

eventId

A unique event ID (string).

timestamp

The UNIX timestamp of when the event occurred.

operation

The operation that triggered the event. Valid values are:

- CREATED
- UPDATED
- DELETED

accountId

Your AWS account ID.

thingGroupId

The ID of the thing group being created, updated, or deleted.

thingGroupName

The name of the thing group being created, updated, or deleted.

versionNumber

The version of the thing group. This value is set to 1 when a thing group is created. It is incremented by 1 each time the thing group is updated.

parentGroupName

The name of the parent thing group (if one exists).

parentGroupId

The ID of the parent thing group (if one exists).

description

A description of the thing group.

rootToParentThingGroups

An array of information about the parent thing group. There is one entry for each parent thing group, starting with the parent of the current thing group and continuing until the root thing group has been reached. Each entry contains the thing group name and the thing group ARN.

attributes

A collection of name-value pairs associated with the thing group.

The following is an example of a payload that is published when a thing group is created.

```
{
    "eventType" : "thingGroupEvent",
    "eventId" : "87f8e095-531c-47b3-aab5-5171364d138d",
    "timestamp" : 1502141209886,
    "operation" : "CREATED",
    "accountId" : "123456789012",
    "thingGroupId" : "8f82a106-6b1d-4331-8984-a84db5f6f8cb",
    "thingGroupName" : "MyRootThingGroup",
    "versionNumber" : 1,
    "parentGroupName" : null,
    "parentGroupId" : null,
    "description" : "My root thing group",
    "rootToParentThingGroups" : null,
    "attributes" : {
        "attribute1" : "value1",
        "attribute3" : "value3",
        "attribute2" : "value2"
```

```
    }
}
```

The following is an example of a payload that is published when a thing group is updated.

```
{
    "eventType" : "thingGroupEvent",
    "eventId" : "63609e4c-2720-466c-8cfe-aa1d897c7f03",
    "timestamp" : 1502141982675,
    "operation" : "UPDATED",
    "accountId" : "123456789012",
    "thingGroupId" : "98e8d404-b093-45e9-97f4-fb02aa2ac733",
    "thingGroupName" : "MyChildThingGroup",
    "versionNumber" : 2,
    "parentGroupName" : null,
    "parentGroupId" : null,
    "description" : "Myupdated child thing group",
    "rootToParentThingGroups" : null,
    "attributes" : {
        "attribute1" : "newValue1",
        "attribute3" : "value3",
        "attribute2" : "value2"
    }
}
```

The following is an example of a payload that is published when a thing group is deleted.

```
{
    "eventType" : "thingGroupEvent",
    "eventId" : "c05cd030-5337-42de-a3a3-9e408f26c628",
    "timestamp" : 1502141623543,
    "operation" : "DELETED",
    "accountId" : "123456789012",
    "thingGroupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "thingGroupName" : "MyChildThingGroup",
    "versionNumber" : 2,
    "parentGroupName" : "MyParentThingGroup",
    "parentGroupId" : "8f82a106-6b1d-4331-8984-a84db5f6f8cb",
    "description" : "My child thing group",
    "rootToParentThingGroups" : null,
    "attributes" : {
        "attribute1" : "newValue1",
        "attribute3" : "value3",
        "attribute2" : "value2"
    }
}
```

Thing Group Membership Events

The payload for a `thingGroupMembershipEvent` event contains the following attributes:

eventType

Set to "thingGroupMembershipEvent" for `thingGroupMembershipEvent` events.

eventId

The event ID.

timestamp

The UNIX timestamp for when the event occurred.

operation

`ADDED` when a thing is added to a thing group. `REMOVED` when a thing is removed from a thing group.

accountId

Your AWS account ID.

groupArn

The ARN of the thing group.

groupId

The ID of the group.

thingArn

The ARN of the thing that was added or removed from the thing group.

thingId

The ID of the thing that was added or removed from the thing group.

membershipId

An ID that represents the relationship between the thing and the thing group. This value is generated when you add a thing to a thing group.

The following is an example of a payload that is published when a thing is added to a thing group.

```
{
    "eventType" : "thingGroupMembershipEvent",
    "eventId" : "d684bd5f-6f6e-48e1-950c-766ac7f02fd1",
    "timestamp" : 1502141492627,
    "operation" : "ADDED",
    "accountId" : "123456789012",
    "groupArn" : "arn:aws:iot:ap-northeast-2:123456789012:thinggroup/MyChildThingGroup",
    "groupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "thingArn" : "arn:aws:iot:ap-northeast-2:123456789012:thing/MyThing",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "membershipId" : "8505ebf8-4d32-4286-80e9-c23a4a16bbd8"
}
```

The following is an example of a payload that is published when a thing is removed from a thing group.

```
{
    "eventType" : "thingGroupMembershipEvent",
    "eventId" : "8749197f-7412-453f-8fcd-8ba1861d3bda",
    "timestamp" : 1502141575159,
    "operation" : "REMOVED",
    "accountId" : "123456789012",
    "groupArn" : "arn:aws:iot:ap-northeast-2:123456789012:thinggroup/MyChildThingGroup",
    "groupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "thingArn" : "arn:aws:iot:ap-northeast-2:123456789012:thing/MyThing",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "membershipId" : "8505ebf8-4d32-4286-80e9-c23a4a16bbd8"
}
```

Thing Group Hierarchy Events

The `thingGroupHierarchyEvent` event occurs when a thing group is added as a child of another thing group. The payload for a `thingGroupHierarchyEvent` event contains the following attributes:

eventType

Set to "thingGroupHierarchyEvent" for `thingGroupHierarchyEvent` events.

eventId

The event ID.

timestamp

The UNIX timestamp for when the event occurred.

operation

`ADDED` when a thing is added to a thing group. `REMOVED` when a thing is removed from a thing group.

accountId

Your AWS account ID.

thingGroupId

The ID of the parent thing group.

thingGroupName

The name of the parent thing group.

childGroupId

The ID of the child thing group.

childGroupName

The name of the child thing group.

The following is an example of a payload that is published when a thing group is added as a child of another thing group.

```
{
    "eventType" : "thingGroupHierarchyEvent",
    "eventId" : "264192c7-b573-46ef-ab7b-489fcd47da41",
    "timestamp" : 1502141354583,
    "operation" : "ADDED",
    "accountId" : "123456789012",
    "thingGroupId" : "8f82a106-6b1d-4331-8984-a84db5f6f8cb",
    "thingGroupName" : "MyRootThingGroup",
    "childGroupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "childGroupName" : "MyChildThingGroup"
}
```

The following is an example of a payload that is published when a thing group is removed as child of another thing group.

```
{
    "eventType" : "thingGroupMembershipEvent",
    "eventId" : "8749197f-7412-453f-8fcd-8ba1861d3bda",
    "timestamp" : 1502141575159,
    "operation" : "REMOVED",
    "accountId" : "123456789012",
    "groupArn" : "arn:aws:iot:ap-northeast-2:123456789012:thinggroup/MyChildThingGroup",
    "groupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "thingArn" : "arn:aws:iot:ap-northeast-2:123456789012:thing/MyThing",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "membershipId" : "8505ebf8-4d32-4286-80e9-c23a4a16bbd8"
```

```
}
```

Jobs Events

Jobs publishes to reserved topics on the MQTT protocol when jobs are pending, completed, or canceled, and when a device reports success or failure when executing a job. Devices or management and monitoring applications can keep track of the status of jobs by subscribing to these topics.

job pending

AWS IoT Jobs publishes a message on an MQTT topic when a job is added to or removed from the list of pending job executions for a thing, or there is a change to the order to the jobs on the list:

- `$aws/things/thingName/jobs/notify`

  The message contains the following example payload:

```
{
    "jobs" : {
        "JobExecutionState": [ JobExecutionSummary (p. 281) ... ],
    },
    "timestamp": timestamp,
}
```

- `$aws/things/thingName/jobs/notify-next`

  The message contains the following example payload:

```
{
    "execution" : JobExecutionData (p. 290),
    "timestamp": timestamp,
}
```

job completed/canceled

AWS IoT Jobs publishes a message on an MQTT topic when a job is completed or canceled:

- `$aws/events/job/jobID/completed`
- `$aws/events/job/jobID/canceled`

The message contains the following example payload:

```
{
    "eventType": "job",
    "eventId": "UUID",
    "timestamp": timestamp,
    "operation": "completed|canceled",
    "jobId":  "043",
    "status": "COMPLETED|CANCELED",
    "targets": [
        "arn:aws:iot:us-west-2:123456789012:thing/xxxxxx",
        "arn:aws:iot:us-west-2:123456789012:thing/yyyyyy",
        "arn:aws:iot:us-west-2:123456789012:thing/zzzzzz"
    ],
    "description": "sample description",
    "completedAt": "14889914167084",
    "createdAt": "14889025672199",
    "lastUpdatedAt": "14889734904359",
    "jobProcessDetails": {
        "numberOfCanceledThings": 1,
```

```
        "numberOfSucceededThings": 1,
        "numberOfRejectedThings": 0,
        "numberOfFailedThings": 1,
        "numberOfInProgressThings": 0,
        "numberOfRemovedThings": 0,
        "processingTargets": [
            arn:aws:iot:us-east-1:123456789012:thing/thingOne,
            arn:aws:iot:us-east-1:123456789012:thing/thingTwo,
            arn:aws:iot:us-east-1:123456789012:thinggroup/thingThree
        ]
    }
}
```

job execution terminal status

AWS IoT Jobs publishes a message when a device updates a job execution to terminal status:

- `$aws/events/jobExecution/`*`jobID`*`/succeeded`
- `$aws/events/jobExecution/`*`jobID`*`/failed`
- `$aws/events/jobExecution/`*`jobID`*`/rejected`
- `$aws/events/jobExecution/`*`jobID`*`/canceled`

The message contains the following example payload:

```
{
    "eventType": "jobExecution",
    "eventId": "UUID",
    "timestamp": "14889025672199",
    "operation": "succeeded|failed|rejected|canceled",
    "jobId":  "031",
    "status": "SUCCESS|FAILED|REJECTED|CANCELED",
    "thingName": "myThing",
}
```

AWS IoT SDKs

Contents

The AWS IoT Device SDKs help you to easily and quickly connect your devices to AWS IoT. The AWS IoT Device SDKs include open-source libraries, developer guides with samples, and porting guides so that you can build innovative IoT products or solutions on your choice of hardware platforms.

AWS Mobile SDK for Android

The AWS SDK for Android contains a library, samples, and documentation for developers to build connected mobile applications using AWS. This SDK also includes support for calling AWS IoT APIs. For more information, see the following:

- AWS Mobile SDK for Android on GitHub
- AWS Mobile SDK for Android Readme
- AWS Mobile SDK for Android Samples

Arduino Yún SDK

The AWS IoT Arduino Yún SDK makes it possible for developers to connect their Arduino Yún-compatible boards to AWS IoT. By connecting a device to AWS IoT, users can securely work with the message broker, rules, and thing shadows provided by AWS IoT and with other AWS services like AWS Lambda, Kinesis, and Amazon S3. For more information, see the following:

- Arduino Yún SDK on GitHub
- Arduino Yún SDK Readme

AWS IoT Device SDK for Embedded C

The AWS IoT Device SDK for Embedded C is a collection of C source files that can be used in embedded applications to securely connect to the AWS IoT platform. It includes transport clients, TLS implementations, and examples for their use. It also supports AWS IoT-specific features such as an API to access the Thing Shadows service. It is distributed as source code and is intended to be built into

customer firmware along with application code, other libraries, and RTOS. For more information, see the following:

- AWS IoT Device SDK for Embedded C GitHub
- AWS IoT Device SDK for Embedded C Readme
- AWS IoT Device SDK for Embedded C Porting Guide

AWS IoT C++ Device SDK

The AWS IoT C++ Device SDK allows developers to build connected applications using AWS and the AWS IoT APIs. Specifically, this SDK was designed for devices that are not resource constrained and required advanced features such as message queuing, multi-threading support, and the latest language features. For more information, see the following:

- AWS IoT C++ Device SDK GitHub
- AWS IoT C++ Device SDK Readme

AWS Mobile SDK for iOS

The AWS SDK for iOS is an open-source software development kit, distributed under an Apache Open Source license. The SDK for iOS provides a library, code samples, and documentation to help developers build connected mobile applications using AWS. This SDK also includes support for calling the AWS IoT API.

- AWS SDK for iOS on GitHub
- AWS SDK for iOS Readme
- AWS SDK for iOS Samples

AWS IoT Device SDK for Java

The AWS IoT Device SDK for Java makes it possible for Java developers to access the AWS IoT platform through MQTT or MQTT over the WebSocket protocol. The SDK is built with AWS IoT thing shadow support. You can access thing shadows by using HTTP methods, including GET, UPDATE, and DELETE. The SDK also supports a simplified thing shadow access model, which allows developers to exchange data with thing shadows by just using getter and setter methods, without having to serialize or deserialize any JSON documents. For more information, see the following:

- AWS IoT Device SDK for Java on GitHub
- AWS IoT Device SDK for Java readme

AWS IoT Device SDK for JavaScript

The aws-iot-device-sdk.js package makes it possible for developers to write JavaScript applications that access AWS IoT using MQTT or MQTT over the WebSocket protocol. It can be used in Node.js environments and browser applications. For more information, see the following:

- AWS IoT Device SDK for JavaScript on GitHub
- AWS IoT Device SDK for JavaScript readme

AWS IoT Device SDK for Python

The AWS IoT Device SDK for Python makes it possible for developers to write Python scripts to use their devices to access the AWS IoT platform through MQTT or MQTT over the WebSocket protocol. By connecting their devices to AWS IoT, users can securely work with the message broker, rules, and thing shadows provided by AWS IoT and with other AWS services like AWS Lambda, Kinesis, and Amazon S3, and more.

- AWS IoT Device SDK for Python on GitHub
- AWS IoT Device SDK for Python readme

Monitoring AWS IoT

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS IoT and your AWS solutions. You should collect monitoring data from all parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. Before you start monitoring AWS IoT, you should create a monitoring plan that includes answers to the following questions:

- What are your monitoring goals?
- Which resources will you monitor?
- How often will you monitor these resources?
- Which monitoring tools will you use?
- Who will perform the monitoring tasks?
- Who should be notified when something goes wrong?

The next step is to establish a baseline for normal AWS IoT performance in your environment, by measuring performance at various times and under different load conditions. As you monitor AWS IoT, store historical monitoring data so that you can compare it with current performance data, identify normal performance patterns and performance anomalies, and devise methods to address issues.

For example, if you're using Amazon EC2, you can monitor CPU utilization, disk I/O, and network utilization for your instances. When performance falls outside your established baseline, you might need to reconfigure or optimize the instance to reduce CPU utilization, improve disk I/O, or reduce network traffic.

To establish a baseline you should, at a minimum, monitor the following items:

- PublishIn.Success
- PublishOut.Success
- Subscribe.Success
- Ping.Success
- Connect.Success
- GetThingShadow.Accepted
- UpdateThingShadow.Accepted
- DeleteThingShadow.Accepted
- RulesExecuted

Topics

Monitoring Tools

AWS provides various tools that you can use to monitor AWS IoT. You can configure some of these tools to do the monitoring for you, while some of the tools require manual intervention. We recommend that you automate monitoring tasks as much as possible.

Automated Monitoring Tools

You can use the following automated monitoring tools to watch AWS IoT and report when something is wrong:

- **Amazon CloudWatch Alarms** – Watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon Simple Notification Service (Amazon SNS) topic or Auto Scaling policy. CloudWatch alarms do not invoke actions simply because they are in a particular state; the state must have changed and been maintained for a specified number of periods. For more information, see Monitoring with Amazon CloudWatch (p. 332).
- **Amazon CloudWatch Logs** – Monitor, store, and access your log files from AWS CloudTrail or other sources. For more information, see Monitoring Log Files in the *Amazon CloudWatch User Guide*.
- **Amazon CloudWatch Events** – Match events and route them to one or more target functions or streams to make changes, capture state information, and take corrective action. For more information, see What is Amazon CloudWatch Events in the *Amazon CloudWatch User Guide*.
- **AWS CloudTrail Log Monitoring** – Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs, write log processing applications in Java, and validate that your log files have not changed after delivery by CloudTrail. For more information, see Working with CloudTrail Log Files in the *AWS CloudTrail User Guide*.

Manual Monitoring Tools

Another important part of monitoring AWS IoT involves manually monitoring those items that the CloudWatch alarms don't cover. The AWS IoT, CloudWatch, and other AWS console dashboards provide an at-a-glance view of the state of your AWS environment. We recommend that you also check the log files on AWS IoT.

- AWS IoT dashboard shows:
  - CA certificates
  - Certificates
  - Polices
  - Rules
  - Things
- CloudWatch home page shows:
  - Current alarms and status
  - Graphs of alarms and resources
  - Service health status

  In addition, you can use CloudWatch to do the following:
  - Create customized dashboards to monitor the services you care about
  - Graph metric data to troubleshoot issues and discover trends
  - Search and browse all your AWS resource metrics
  - Create and edit alarms to be notified of problems

Monitoring with Amazon CloudWatch

You can monitor AWS IoT using CloudWatch, which collects and processes raw data from AWS IoT into readable, near real-time metrics. These statistics are recorded for a period of two weeks, so that you can access historical information and gain a better perspective on how your web application or service is

performing. By default, AWS IoT metric data is automatically sent to CloudWatch in 1 minute periods. For more information, see What Are Amazon CloudWatch, Amazon CloudWatch Events, and Amazon CloudWatch Logs? in the *Amazon CloudWatch User Guide*.

Topics

AWS IoT Metrics and Dimensions

When you interact with AWS IoT, it sends the following metrics and dimensions to CloudWatch every minute. You can use the following procedures to view the metrics for AWS IoT.

To view metrics using the CloudWatch console

Metrics are grouped first by the service namespace, and then by the various dimension combinations within each namespace.

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. In the **CloudWatch Metrics by Category** pane, under the metrics category for AWS IoT, select a metrics category, and then in the upper pane, scroll down to view the full list of metrics.

To view metrics using the AWS CLI

- At a command prompt, use the following command:

```
aws cloudwatch list-metrics --namespace "AWS/IoT"
```

CloudWatch displays the following metrics for AWS IoT:

AWS IoT Metrics

AWS IoT sends the following metrics to CloudWatch once per received request.

IoT Metrics

Metric	Description
RulesExecuted	The number of AWS IoT rules executed.

Rule Metrics

Metric	Description
TopicMatch	The number of incoming messages published on a topic on which a rule is listening. The `RuleName` dimension contains the name of the rule.
ParseError	The number of JSON parse errors that occurred in messages published on a topic on which a rule is

Metric	Description
	listening. The `RuleName` dimension contains the name of the rule.

Rule Action Metrics

Metric	Description
Success	The number of successful rule action invocations. The `RuleName` dimension contains the name of the rule that specifies the action. The `ActionType` dimension contains the type of action that was invoked.
Failure	The number of failed rule action invocations. The `RuleName` dimension contains the name of the rule that specifies the action. The `RuleName` dimension contains the name of the rule that specifies the action. The `ActionType` dimension contains the type of action that was invoked.

Message Broker Metrics

Metric	Description
Connect.AuthError	The number of connection requests that could not be authorized by the message broker. The `Protocol` dimension contains the protocol used to send the `CONNECT`. message.
Connect.ClientError	The number of connection requests rejected because the MQTT message did not meet the requirements defined in AWS IoT Limits. The `Protocol` dimension contains the protocol used to send the `CONNECT`. message.
Connect.ServerError	The number of connection requests that failed because an internal error occurred. The Protocol dimension contains the protocol used to send the `CONNECT` message.
Connect.Success	The number of successful connections to the message broker. The `Protocol` dimension contains the protocol used to send the `CONNECT` message.
Connect.Throttle	The number of connection requests that were throttled because the client exceeded the allowed connect request rate. The `Protocol` dimension contains the protocol used to send the `CONNECT` message.
Ping.Success	The number of ping messages received by the message broker. The `Protocol` dimension contains the protocol used to send the ping message.
PublishIn.AuthError	The number of publish requests the message broker was unable to authorize. The `Protocol` dimension contains the protocol used to publish the message.

Metric	Description
PublishIn.ClientError	The number of publish requests rejected by the message broker because the message did not meet the requirements defined in AWS IoT Limits. The `Protocol` dimension contains the protocol used to publish the message.
PublishIn.ServerError	The number of publish requests the message broker failed to process because an internal error occurred. The `Protocol` dimension contains the protocol used to send the `PUBLISH` message.
PublishIn.Success	The number of publish requests successfully processed by the message broker. The `Protocol` dimension contains the protocol used to send the `PUBLISH` message.
PublishIn.Throttle	The number of publish request that were throttled because the client exceeded the allowed inbound message rate. The `Protocol` dimension contains the protocol used to send the `PUBLISH` message.
PublishOut.AuthError	The number of publish requests made by the message broker that could not be authorized by AWS IoT. The `Protocol` dimension contains the protocol used to send the `PUBLISH` message.
PublishOut.ClientError	The number of publish requests made by the message broker that were rejected because the message did not meet the requirements defined in AWS IoT Limits. The `Protocol` dimension contains the protocol used to send the `PUBLISH` message.
PublishOut.Success	The number of publish requests successfully made by the message broker. The `Protocol` dimension contains the protocol used to send the `PUBLISH` message.
Subscribe.AuthError	The number of subscription requests made by a client that could not be authorized. The `Protocol` dimension contains the protocol used to send the `SUBSCRIBE` message.
Subscribe.ClientError	The number of subscribe requests that were rejected because the `SUBSCRIBE` message did not meet the requirements defined in AWS IoT Limits. The `Protocol` dimension contains the protocol used to send the `SUBSCRIBE` message.
Subscribe.ServerError	The number of subscribe requests that were rejected because an internal error occurred. The `Protocol` dimension contains the protocol used to send the `SUBSCRIBE` message.
Subscribe.Success	The number of subscribe requests that were successfully processed by the message broker. The `Protocol` dimension contains the protocol used to send the `SUBSCRIBE` message.

Metric	Description
Subscribe.Throttle	The number of subscribe requests that were throttled because the client exceeded the allowed subscribe request rate. The `Protocol` dimension contains the protocol used to send the `SUBSCRIBE` message.
Unsubscribe.ClientError	The number of unsubscribe requests that were rejected because the `UNSUBSCRIBE` message did not meet the requirements defined in AWS IoT Limits. The `Protocol` dimension contains the protocol used to send the `UNSUBSCRIBE` message.
Unsubscribe.ServerError	The number of unsubscribe requests that were rejected because an internal error occurred. The `Protocol` dimension contains the protocol used to send the `UNSUBSCRIBE` message.
Unsubscribe.Success	The number of unsubscribe requests that were successfully processed by the message broker. The `Protocol` dimension contains the protocol used to send the `UNSUBSCRIBE` message.
Unsubscribe.Throttle	The number of unsubscribe requests that were rejected because the client exceeded the allowed unsubscribe request rate. The `Protocol` dimension contains the protocol used to send the `UNSUBSCRIBE` message.

Note
The message broker metrics are displayed in the AWS IoT console under **Protocol Metrics**.

Thing Shadow Metrics

Metric	Description
DeleteThingShadow.Accepted	The number of DeleteThingShadow requests processed successfully. The `Protocol` dimension contains the protocol used to make the request.
GetThingShadow.Accepted	The number of GetThingShadow requests processed successfully. The `Protocol` dimension contains the protocol used to make the request.
UpdateThingShadow.Accepted	The number of UpdateThingShadow requests processed successfully. The `Protocol` dimension contains the protocol used to make the request.

Note
The thing shadow metrics are displayed in the AWS IoT console under **Protocol Metrics**.

Dimensions for Metrics

Metrics use the namespace and provide metrics for the following dimension(s):

Dimension	Description
ActionType	The action type specified by the rule that triggered by the request.
Protocol	The protocol used to make the request. Valid values are: MQTT or HTTP
RuleName	The name of the rule triggered by the request.

How Do I Use AWS IoT Metrics?

The metrics reported by AWS IoT provide information that you can analyze in different ways. The following use cases are based on a scenario where you have ten things that connect to the internet once a day. Each day:

- Ten things connect to AWS IoT at roughly the same time.
- Each thing subscribes to a topic filter, and then waits for an hour before disconnecting. During this period, things communicate with one another and learn more about the state of the world.
- Each thing publishes some perception it has based on its newly found data using `UpdateThingShadow`.
- Each thing disconnects from AWS IoT.

These are suggestions to get you started, not a comprehensive list.

- How can I be notified if my things do not connect successfully each day? (p. 337)
- How can I be notified if my things are not publishing data each day? (p. 338)
- How can I be notified if my thing's shadow updates are being rejected each day? (p. 339)

Creating CloudWatch Alarms to Monitor AWS IoT

You can create a CloudWatch alarm that sends an Amazon SNS message when the alarm changes state. An alarm watches a single metric over a time period you specify and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon SNS topic or Auto Scaling policy. Alarms invoke actions for sustained state changes only. CloudWatch alarms do not invoke actions simply because they are in a particular state; the state must have changed and been maintained for a specified number of periods.

How can I be notified if my things do not connect successfully each day?

1. Create an Amazon SNS topic, arn:aws:sns:us-east-1:123456789012:things-not-connecting-successfully.

   For more information, see Set Up Amazon Simple Notification Service.

2. Create the alarm.

```
Prompt>aws cloudwatch put-metric-alarm \
    --alarm-name ConnectSuccessAlarm \
    --alarm-description "Alarm when my Things don't connect successfully" \
```

```
    --namespace AWS/IoT \
    --metric-name Connect.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:us-east-1:1234567890:things-not-connecting-successfully
```

```
Prompt>aws cloudwatch put-metric-alarm \
    --alarm-name ConnectSuccessAlarm \
    --alarm-description "Alarm when my Things don't connect successfully" \
    --namespace AWS/IoT \
    --metric-name Connect.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:us-east-1:1234567890:things-not-connecting-successfully
```

3. Test the alarm.

```
Prompt>aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --state-reason
 "initializing" --state-value OK
```

```
 Prompt>aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --state-reason
 "initializing" --state-value ALARM
```

How can I be notified if my things are not publishing data each day?

1. Create an Amazon SNS topic, `arn:aws:sns:us-east-1:123456789012:things-not-publishing-data`.

   For more information, see Set Up Amazon Simple Notification Service.

2. Create the alarm.

```
Prompt>aws cloudwatch put-metric-alarm \
    --alarm-name PublishInSuccessAlarm\
    --alarm-description "Alarm when my Things don't publish their data \
    --namespace AWS/IoT \
    --metric-name PublishIn.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:us-east-1:1234567890:things-not-publishing-data
```

3. Test the alarm.

```
Prompt>aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --state-reason
 "initializing" --state-value OK
```

```
Prompt>aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --state-reason
 "initializing" --state-value ALARM
```

How can I be notified if my thing's shadow updates are being rejected each day?

1. Create an Amazon SNS topic, arn:aws:sns:us-east-1:1234567890:things-shadow-updates-rejected.

   For more information, see Set Up Amazon Simple Notification Service.
2. Create the alarm.

```
Prompt>aws cloudwatch put-metric-alarm \
    --alarm-name UpdateThingShadowSuccessAlarm \
    --alarm-description "Alarm when my Things Shadow updates are getting rejected" \
    --namespace AWS/IoT \
    --metric-name UpdateThingShadow.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:us-east-1:1234567890:things-shadow-updates-rejected
```

3. Test the alarm.

```
Prompt>aws cloudwatch set-alarm-state --alarm-name UpdateThingShadowSuccessAlarm --
state-reason "initializing" --state-value OK
```

```
Prompt>aws cloudwatch set-alarm-state --alarm-name UpdateThingShadowSuccessAlarm --
state-reason "initializing" --state-value ALARM
```

Logging AWS IoT API Calls with AWS CloudTrail

AWS IoT is integrated with CloudTrail, a service that captures all of the AWS IoT API calls and delivers the log files to an Amazon S3 bucket that you specify. CloudTrail captures API calls from the AWS IoT console or from your code to the AWS IoT APIs. Using the information collected by CloudTrail, you can determine the request that was made to AWS IoT, the source IP address from which the request was made, who made the request, when it was made, and so on.

To learn more about CloudTrail, including how to configure and enable it, see the *AWS CloudTrail User Guide*.

AWS IoT Information in CloudTrail

When CloudTrail logging is enabled in your AWS account, API calls made to AWS IoT actions are tracked in CloudTrail log files where they are written with other AWS service records. CloudTrail determines when to create and write to a new file based on a time period and file size.

All AWS IoT actions are logged by CloudTrail and are documented in the AWS IoT API Reference. For example, calls to the **CreateThing**, **ListThings**, and **ListTopicRules** sections generate entries in the CloudTrail log files.

Every log entry contains information about who generated the request. The user identity information in the log entry helps you determine the following:

- Whether the request was made with root or 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.

You can store your log files in your Amazon S3 bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted with Amazon S3 server-side encryption (SSE).

If you want to be notified upon log file delivery, you can configure CloudTrail to publish Amazon SNS notifications when new log files are delivered. For more information, see Configuring Amazon SNS Notifications for CloudTrail.

You can also aggregate AWS IoT log files from multiple AWS regions and multiple AWS accounts into a single Amazon S3 bucket.

For more information, see Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts.

Understanding AWS IoT Log File Entries

CloudTrail log files can contain one or more log entries. Each entry lists multiple JSON-formatted events. A log entry represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. Log entries are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `AttachPolicy` action.

```
{
    "timestamp":"1460159496",
    "AdditionalEventData":"",
    "Annotation":"",
    "ApiVersion":"",
    "ErrorCode":"",
    "ErrorMessage":"",
    "EventID":"8bff4fed-c229-4d2d-8264-4ab28a487505",
    "EventName":"AttachPolicy",
    "EventTime":"2016-04-08T23:51:36Z",
    "EventType":"AwsApiCall",
    "ReadOnly":"",
    "RecipientAccountList":"",
    "RequestID":"d4875df2-fde4-11e5-b829-23bf9b56cbcd",
    "RequestParamters":{
        "principal":"arn:aws:iot:us-
east-1:123456789012:cert/528ce36e8047f6a75ee51ab7beddb4eb268ad41d2ea881a10b67e8e76924d894",
        "policyName":"ExamplePolicyForIoT"
    },
    "Resources":"",
    "ResponseElements":"",
    "SourceIpAddress":"52.90.213.26",
```

```
    "UserAgent":"aws-internal/3",
    "UserIdentity":{
        "type":"AssumedRole",
        "principalId":"AKIAI44QH8DHBEXAMPLE",
        "arn":"arn:aws:sts::12345678912:assumed-role/iotmonitor-us-east-1-beta-
InstanceRole-1C5T1YCYMHPYT/i-35d0a4b6",
        "accountId":"222222222222",
        "accessKeyId":"access-key-id",
        "sessionContext":{
            "attributes":{
                "mfaAuthenticated":"false",
                "creationDate":"Fri Apr 08 23:51:10 UTC 2016"
            },
            "sessionIssuer":{
                "type":"Role",
                "principalId":"AKIAI44QH8DHBEXAMPLE",
                "arn":"arn:aws:iam::123456789012:role/executionServiceEC2Role/iotmonitor-
us-east-1-beta-InstanceRole-1C5T1YCYMHPYT",
                "accountId":"222222222222",
                "userName":"iotmonitor-us-east-1-InstanceRole-1C5T1YCYMHPYT"
            }
        },
        "invokedBy":{
            "serviceAccountId":"111111111111"
        }
    },
    "VpcEndpointId":""
}
```

Troubleshooting AWS IoT

The following information might help you troubleshoot common issues in AWS IoT.

Tasks

Diagnosing Connectivity Issues

Authentication

How do my devices authenticate AWS IoT endpoints?

Add the AWS IoT CA certificate to your client's trust store. You can download the CA certificate from here.

How can I validate a correctly configured certificate?

Use the OpenSSL `s_client` command to test a connection to the AWS IoT endpoint:

```
openssl s_client -connect custom_endpoint.iot.us-east-1.amazonaws.com:8443 -
CAfile CA.pem -cert cert.pem -key privateKey.pem
```

Authorization

I received a `PUBNACK` or `SUBNACK` response from the broker. What do I do?

Make sure that there is a policy attached to the certificate you are using to call AWS IoT. All publish/subscribe operations are denied by default.

Setting Up CloudWatch Logs with AWS IoT

AWS IoT sends progress events about each message as it passes from your devices through the message broker and the rules engine. To view these logs, you must configure AWS IoT to generate the logs used by CloudWatch.

For more information about CloudWatch Logs, see CloudWatch Logs.

To enable AWS IoT logging, you must create an IAM role, register the role with AWS IoT, and then configure AWS IoT logging.

Note
Before you enable AWS IoT logging, make sure you understand the CloudWatch Logs access permissions. Users with access to CloudWatch Logs can see debugging information from your devices. For more information see Authentication and Access Control for Amazon CloudWatch Logs.

Create a Logging Role

Use the IAM console to create a logging role.

1. From the navigation pane, choose **Roles**, and then choose **Create new role**.
2. Choose **AWS Service Role** and for service role type, choose **AWS IoT**.
3. Choose the **AWSIoTLogging** role, and then choose **Next Step**.
4. Type a name and description for the role, and then choose **Create role**.

Logging Role Policy

The following policy documents provide the role policy and trust policy that allow AWS IoT to submit logs to CloudWatch on your behalf.

Note
These documents were created for you when you created the logging role.

Role policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy"
             ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Trust policy:

```
{
 "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

Log Level

The log level specifies which types of logs are generated.

ERROR

Any error that causes an operation to fail.

Logs include ERROR information only.

WARN

Anything that can potentially cause inconsistencies in the system, but might not cause the operation to fail.

Logs include ERROR and WARN information.

INFO

High-level information about the flow of things.

Logs include INFO, ERROR, and WARN information.

DEBUG

Information that might be helpful when debugging a problem.

Logs include DEBUG, INFO, ERROR, and WARN information.

DISABLED

All logging is disabled.

Configure AWS IoT Logging

You can configure logging in two ways, global logging and fine grained logging. Global logging sets one logging level for all logs no matter what resource triggered the logs. Fine-grained logging allows you to set a logging level for a specific resource or set of resources. Currently only thing groups are supported.

Global Logging

Use the `set-logging-options` CLI command to set the logging options for your account. `set-logging-options` takes two arguments:

`roleArn`

Your logging role ARN. The logging role grants AWS IoT permission to write to your CloudWatch Logs.

`logLevel`

The log level to use. Valid values are: ERROR, WARN, INFO, DEBUG, or DISABLED

For example:

```
aws iot set-logging-options
                        --logging-options-payload
                            roleArn="arn:aws:iam::<your-aws-account-
num>:role/<IoTLoggingRole>",logLevel="<INFO>"
```

You can use the `get-logging-options` CLI command to get the current logging options.

Fine-Grained Logging

Fine-grained logging allows you to specify a logging level for a target. A target is defined by a resource type and a resource name. Currently, AWS IoT supports thing groups as targets. Fine-grained logging allows you to set a logging level for a specific thing group. Say we have a thing group called "Phones" that contains things that represent different kinds of phones. We then create another thing group called "MobilePhones" and make it a child of the "Phones" thing group. Fine-grained logging allows you to configure one logging level for all things in the "Phones" group (and any child groups) and another logging level for things in the "MobilePhones" group. In this example, we have assigned two different logging levels to things in the "MobilePhones" group — one from the logging level for the "Phones" thing group and another from the "MobilePhones" thing group — but the logging level specified for the child thing group will override the logging level specified for the parent thing group.

Use the `set-v2-logging-options` CLI command to enable fine-grained logging and set the default logging level. It takes the following optional arguments:

--role-arn

An IAM role that allows AWS IoT to write to your CloudWatch Logs. If not specified, AWS IoT uses the logging role associated with your account. The logging role is associated with your account when it is created. For more information, see Create a Logging Role (p. 343)

--default-log-level

The logging level used if not specified. Valid values are: `DEBUG`, `INFO`, `ERROR`, `WARN`, and `DISABLED`

--disable-all-logging

If set to `true`, all logging is disabled.

The `get-v2-logging-options` CLI command returns the configured IAM logging role, the default logging level, and the `disableAllLogs` value.

Use the `set-v2-logging-level` CLI command to configure fine-grained logging for a target. It takes the following arguments:

--log-target

A JSON document that contains the resource type and name of the entity for which you are configuring logging. AWS IoT currently supports `THING_GROUP` for resource type. You can configure up to 10 logging targets.

--log-level

The logging level used when generating logs for the specified resource. Valid values are: `DEBUG`, `INFO`, `ERROR`, `WARN`, and `DISABLED`

Use the `list-v2-logging-levels` CLI command to get a list of the currently configured fine-grained logging levels. Call the `delete-v2-logging-level` CLI command to delete a logging level. Use the `delete-v2-logging-level` command to delete a fine-grained logging level.

CloudWatch Log Entry Format

Each component of AWS IoT generates its own logs. Each log entry has an `eventType` that indicates which operation caused the log to be generated. This section describes the logs generated by the following AWS IoT components:

- Message Broker (p. 346)

- Thing Shadows (p. 350)
- Rules Engine (p. 352)
- Jobs (p. 355)

All CloudWatch Logs have the following common attributes:

timestamp

The UNIX timestamp of when the client connected to the AWS IoT message broker.

logLevel

The log level being used. For more information, see the section called "Log Level" (p. 344).

traceId

A randomly generated identifier that can be used to correlate all logs for a specific request.

accountId

Your AWS account ID.

status

The status of the request.

eventType

The event type for which the log was generated. The value of the event type for each event is listed in the following sections.

Message Broker Logs

The AWS IoT message broker generates logs for the following events:

Connect Log

The AWS IoT message broker generates a `Connect` log when an MQTT client connects.

more info (1)

For example:

```
{
    "timestamp": "2017-08-10 15:37:23.476",
 "logLevel": "INFO",
 "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "Connect",
 "protocol": "MQTT",
 "clientId": "abf27092886e49a8a5c1922749736453",
 "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
 "sourceIp": "205.251.233.181",
 "sourcePort": 13490
}
```

In addition to the attributes common to all CloudWatch Logs, `Connect` log entries contain the following attributes:

eventType

`Connect` for connection logs.

protocol

The protocol used when making the request. Valid values are `MQTT` or `HTTP`.

clientId

The ID of the client making the request.

principalId

The ID of the principal making the request.

sourceIp

The IP address where the request originated.

sourcePort

The port where the request originated.

Subscribe Log

The AWS IoT message broker generates a `Subscribe` log when an MQTT client subscribes to a topic. more info (2)

For example:

```
{
 "timestamp": "2017-08-10 15:39:04.413",
 "logLevel": "INFO",
 "traceId": "7aa5c38d-1b49-3753-15dc-513ce4ab9fa6",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "Subscribe",
 "protocol": "MQTT",
 "topicName": "$aws/things/MyThing/shadow/#",
 "clientId": "abf27092886e49a8a5c1922749736453",
 "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
 "sourceIp": "205.251.233.181",
 "sourcePort": 13490
}
```

In addition to the attributes common to all CloudWatch Logs, `Subscribe` log entries contain the following attributes:

eventType

`Subscribe` for subscription logs.

protocol

The protocol used when making the request. Valid values are `MQTT` or `HTTP`.

topicName

The name of the subscribed topic.

clientId

The ID of the client making the request.

principalId

The ID of the principal making the request.

sourceIp

The IP address where the request originated.

sourcePort

The port where the request originated.

Publish-In Log

The AWS IoT message broker generates a `Publish-In` log when the AWS IoT message broker receives an MQTT message.

more info (3)

For example:

```
{
 "timestamp": "2017-08-10 15:39:30.961",
 "logLevel": "INFO",
 "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "Publish-In",
 "protocol": "MQTT",
 "topicName": "$aws/things/MyThing/shadow/get",
 "clientId": "abf27092886e49a8a5c1922749736453",
 "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
 "sourceIp": "205.251.233.181",
 "sourcePort": 13490
}
```

In addition to the attributes common to all CloudWatch Logs, `Publish-In` log entries contain the following attributes:

eventType

`Publish-In` when the message broker receives a message.

status

The status of the request.

protocol

The protocol used when making the request. Valid values are `MQTT` or `HTTP`.

topicName

The name of the subscribed topic.

clientId

The ID of the client making the request.

principalId

The ID of the principal making the request.

sourceIp

The IP address where the request originated.

sourcePort

The port where the request originated.

Publish-Out Log

The AWS IoT message broker generates a `Publish-Out` log when the message broker publishes an MQTT message.

more info (4)

For example:

```
{
 "timestamp": "2017-08-10 15:39:30.961",
 "logLevel": "INFO",
 "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "Publish-Out",
 "protocol": "MQTT",
 "topicName": "$aws/things/MyThing/shadow/get",
 "clientId": "abf27092886e49a8a5c1922749736453",
 "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
 "sourceIp": "205.251.233.181",
 "sourcePort": 13490
}
```

In addition to the attributes common to all CloudWatch Logs, `Publish-Out` log entries contain the following attributes:

eventType

`Publish-Out` when the message broker publishes a message.

status

The status of the request.

protocol

The protocol used when making the request. Valid values are `MQTT` or `HTTP`.

topicName

The name of the subscribed topic.

clientId

The ID of the client making the request.

principalId

The ID of the principal making the request.

sourceIp

The IP address where the request originated.

sourcePort

The port where the request originated.

Disconnect Log

The AWS IoT message broker generates a `Disconnect` log when an MQTT client disconnects.

more info (5)

For example:

```
{
    "timestamp": "2017-08-10 15:37:23.476",
 "logLevel": "INFO",
 "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "Disconnect",
 "protocol": "MQTT",
 "clientId": "abf27092886e49a8a5c1922749736453",
 "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
 "sourceIp": "205.251.233.181",
 "sourcePort": 13490
}
```

In addition to the attributes common to all CloudWatch Logs, `Disconnect` log entries contain the following attributes:

eventType

`Disconnect` for connection logs.

protocol

The protocol used when making the request. Valid values are `MQTT` or `HTTP`.

clientId

The ID of the client making the request.

principalId

The ID of the principal making the request.

sourceIp

The IP address where the request originated.

sourcePort

The port where the request originated.

Thing Shadow Logs

The AWS IoT Thing Shadow service generates logs for the following events:

GetThingShadow Logs

The Thing Shadow service generates a `GetThingShadow` log when a get request for a thing shadow is received.

more info (6)

For example:

```
{
 "timestamp": "2017-08-09 17:56:30.941",
 "logLevel": "INFO",
 "traceId": "b575f19a-97a2-cf72-0ed0-c64a783a2504",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "GetThingShadow",
 "protocol": "MQTT",
 "deviceShadowName": "MyThing",
 "topicName": "$aws/things/MyThing/shadow/get"
```

```
}
```

In addition to the attributes common to all CloudWatch Logs, `GetThingShadow` log entries contain the following attributes:

eventType

`GetThingShadow` for GetThingShadow logs.

protocol

The protocol used when making the request. Valid values are `MQTT` or `HTTP`.

deviceShadowName

The name of the requested thing shadow.

topicName

The name of the topic on which the request was published.

UpdateThingShadow Logs

The Thing Shadow service generates a `UpdateThingShadow` log when a request to update a thing shadow is received.

more info (7)

For example:

```
{
 "timestamp": "2017-08-07 18:43:59.436",
 "logLevel": "INFO",
 "traceId": "d0074ba8-0c4b-a400-69df-76326d414c28",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "UpdateThingShadow",
 "protocol": "MQTT",
 "deviceShadowName": "Jack",
  "topicName": "$aws/things/Jack/shadow/update"
}
```

In addition to the attributes common to all CloudWatch Logs, `UpdateThingShadow` log entries contain the following attributes:

eventType

`UpdateThingShadow` for update shadow logs.

protocol

The protocol used when making the request. Valid values are `MQTT` or `HTTP`.

deviceShadowName

The name of the thing shadow to update.

topicName

The name of the topic on which the request was published.

DeleteThingShadow Logs

The Thing Shadow service generates a `DeleteThingShadow` log when a request to delete a thing shadow is received.

more info (8)

For example:

```
{
 "timestamp": "2017-08-07 18:47:56.664",
 "logLevel": "INFO",
 "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "DeleteThingShadow",
 "protocol": "MQTT",
 "deviceShadowName": "Jack",
 "topicName": "$aws/things/Jack/shadow/delete"
}
```

In addition to the attributes common to all CloudWatch Logs, `DeleteThingShadow` log entries contain the following attributes:

eventType

`DeleteThingShadow` for DeleteThingShadow logs.

protocol

The protocol used when making the request. Valid values are `MQTT` or `HTTP`.

deviceShadowName

The name of the thing shadow to update.

topicName

The name of the topic on which the request was published.

Rules Engine Logs

The AWS IoT Rules Engine service generates logs for the following events:

Rule Match Logs

The AWS IoT rules engine generates a `RuleMatch` log when the message broker receives a message that matches a rule.

more info (9)

For example:

```
{
 "timestamp": "2017-08-10 16:32:46.002",
 "logLevel": "INFO",
 "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "RuleMatch",
 "clientId": "abf27092886e49a8a5c1922749736453",
 "topicName": "rules/test",
 "ruleName": "JSONLogsRule",
 "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

In addition to the attributes common to all CloudWatch Logs, `RuleMatch` log entries contain the following attributes:

eventType

`RuleMatch` for rule match logs.

clientId

The ID of the client making the request.

topicName

The name of the subscribed topic.

ruleName

The name of the matching rule.

principalId

The ID of the principal making the request.

Function Execution Logs

The rules engine generates a `FunctionExecution` log when a rule's SQL query calls an external function. An external function is called when a rule's action makes an HTTP request to AWS IoT or another web service (for example, calling `get_thing_shadow` or `machinelearning_predict`).

more info (10)

A `FunctionExecution` log will look like the following:

```
{
    "timestamp": "2017-07-13 18:33:51.903”,
    "logLevel": "DEBUG”,
    "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5”,
    "status": "Success”,
    "eventType": "FunctionExecution”,
    "clientId": "N/A”,
    "topicName":"rules/test”,
    "ruleName": "ruleTestPredict”,
    "ruleAction": "MachinelearningPredict”,
    "resources": {
        "ModelId": "predict-model”
    },
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167”
}
```

In addition to the attributes common to all CloudWatch Logs, `FunctionExecution` log entries contain the following attributes:

eventType

`FunctionExecution` for rule match logs.

clientId

`N/A` for `FunctionExecution` logs.

topicName

The name of the subscribed topic.

ruleName

The name of the matching rule.

resources

A collection of resources used by the rule's actions.

principalId

The ID of the principal making the request.

Starting Execution Logs

The AWS IoT rules engine generates a `StartingExecution` log when the rules engine starts to invoke a rule's action.

more info (11)

For example:

```
{
 "timestamp": "2017-08-10 16:32:46.002",
 "logLevel": "DEBUG",
 "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "StartingRuleExecution",
 "clientId": "abf27092886e49a8a5c1922749736453",
 "topicName": "rules/test",
 "ruleName": "JSONLogsRule",
 "ruleAction": "RepublishAction",
 "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

In addition to the attributes common to all CloudWatch Logs, `StartingExecution` log entries contain the following attributes:

eventType

`StartingRuleExecution` for starting rule execution logs.

clientId

The ID of the client making the request.

topicName

The name of the subscribed topic.

ruleName

The name of the matching rule.

ruleAction

The name of the invoked action.

principalId

The ID of the principal making the request.

Rule Execution Logs

The AWS IoT rules engine generates a `RuleExecution` log when the rules engine invokes a rule's action.

more info (12)

For example:

```
{
```

```
 "timestamp": "2017-08-10 16:32:46.070",
 "logLevel": "INFO",
 "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "RuleExecution",
 "clientId": "abf27092886e49a8a5c1922749736453",
 "topicName": "rules/test",
 "ruleName": "JSONLogsRule",
 "ruleAction": "RepublishAction",
 "resources": {
  "RepublishTopic": "rules/republish"
 },
 "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

In addition to the attributes common to all CloudWatch Logs, `RuleExecution` log entries contain the following attributes:

eventType

`RuleExecution` for rule execution logs.

clientId

The ID of the client making the request.

topicName

The name of the subscribed topic.

ruleName

The name of the matching rule.

ruleAction

The name of the invoked action.

resources

A collection of resources used by the rule's actions.

principalId

The ID of the principal making the request.

Job Logs

The AWS IoT Job service generates logs for the following events:

Get Job Execution Logs

The AWS IoT rules engine generates a `GetJobExecution` log when the Job service receives a request for a job execution.

more info (13)

For example:

```
{
 "timestamp": "2017-08-10 19:08:05.813",
 "logLevel": "DEBUG",
 "accountId": "123456789012",
```

```
 "status": "Success",
 "eventType": "GetJobExecution",
 "protocol": "MQTT",
 "clientId": "thingOne",
 "topicName": "$aws/things/thingOne/jobs/get",
 "details": "The request status is SUCCESS."
}
```

In addition to the attributes common to all CloudWatch Logs, `GetJobExecution` log entries contain the following attributes:

eventType

`GetJobExecution` for get job execution logs.

protocol

The protocol used when making the request. Valid values are `MQTT` or `HTTP`.

clientId

The ID of the client making the request.

topicName

The name of the subscribed topic.

details

Additional information from the Jobs service.

Describe Job Execution Logs

The AWS IoT rules engine generates a `DescribeJobExecution` log when the Jobs service receives a request to describe a job execution.

more info (14)

For example:

```
{
 "timestamp": "2017-08-10 19:13:22.841",
 "logLevel": "DEBUG",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "DescribeJobExecution",
 "protocol": "MQTT",
 "clientId": "thingOne",
 "jobId": "002",
 "topicName": "$aws/things/thingOne/jobs/002/get",
 "clientToken": "myToken",
 "details": "The request status is SUCCESS."
}
```

In addition to the attributes common to all CloudWatch Logs, `GetJobExecution` log entries contain the following attributes:

eventType

`DescribeJobExecution` for describe job execution logs.

protocol

The protocol used when making the request. Valid values are `MQTT` or `HTTP`.

clientId

The ID of the client making the request.

jobId

The job ID for the job execution.

topicName

The topic used to make the request.

clientToken

The client token.

details

Additional information from the Jobs service.

Update Job Execution Logs

The AWS IoT rules engine generates an `UpdateJobExecution` log when the Jobs service receives a request to update a job execution.

more info (15)

For example:

```
{
 "timestamp": "2017-08-10 19:25:14.758",
 "logLevel": "DEBUG",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "UpdateJobExecution",
 "protocol": "MQTT",
 "clientId": "thingOne",
 "jobId": "002",
 "topicName": "$aws/things/thingOne/jobs/002/update",
 "clientToken": "myClientToken",
 "versionNumber": "1",
 "details": "The destination status is IN_PROGRESS. The request status is SUCCESS."
}
```

In addition to the attributes common to all CloudWatch Logs, `UpdateJobExecution` log entries contain the following attributes:

eventType

`UpdateJobExecution` for update job execution logs.

protocol

The protocol used when making the request. Valid values are `MQTT` or `HTTP`.

clientId

The ID of the client making the request.

jobId

The job ID for the job execution.

topicName

The topic used to make the request.

clientToken

The client token.

versionNumber

The version of the job execution.

details

Additional information from the Jobs service.

Report Final Job Execution Count

The AWS IoT Jobs service generates a `ReportFinalJobExecutionCount` log when a job is completed.

more info (16)

For example:

```
{
 "timestamp": "2017-08-10 19:44:16.776",
 "logLevel": "INFO",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "ReportFinalJobExecutionCount",
 "protocol": "MQTT",
 "jobId": "002",
 "details": "Job 002 completed. QUEUED job execution count: 0 IN_PROGRESS job execution
 count: 0 FAILED job execution count: 0 SUCCESS job execution count: 1 CANCELED job
 execution count: 0 REJECTED job execution count: 0 REMOVED job execution count: 0"
}
```

In addition to the attributes common to all CloudWatch Logs, `ReportFinalJobExecutionCount` log entries contain the following attributes:

eventType

`ReportFinalJobExecutionCount` for report final job execution count logs.

protocol

The protocol used when making the request. Valid values are `MQTT` or `HTTP`.

jobId

The job ID for the job execution.

details

Additional information from the Jobs service.

Viewing Logs

To view your logs

1. Browse to https://console.aws.amazon.com/cloudwatch/. In the navigation pane, choose **Logs**.
2. In the **Filter** text box, type **`AWSIotLogsV2`** , and press Enter.
3. Double-click the `AWSIotLogsV2` log group.
4. Choose **Search Log Group**. A complete list of the AWS IoT logs generated for your account is displayed.

5. Choose the expand icon to look at an individual stream.

You can also type a query in the **Filter events** text box. Here are some interesting queries to try:

- `{ $.logLevel = "INFO" }`

  Find all logs that have a log level of `INFO`.
- `{ $.status = "Success" }`

  Find all logs that have a status of `Success`.
- `{ $.status = "Success" && $.eventType = "GetThingShadow" }`

  Find all logs that have a status of `Success` and an event type of `GetThingShadow`.

For more information about creating filter expressions, see CloudWatch Logs Queries.

Diagnosing Rules Issues

CloudWatch Logs is the best place to debug issues you are having with rules. When you enable CloudWatch Logs for AWS IoT, you can see which rules are triggered and their success or failure. You also get information about whether WHERE clause conditions match.

The most common rules issue is authorization. The logs show if your role is not authorized to perform AssumeRole on the resource. Here is an example log generated by fine-grained logging (p. 345):

```
{
    "timestamp": "2017-12-09 22:49:17.954",
    "logLevel": "ERROR",
    "traceId": "ff563525-6469-506a-e141-78d40375fc4e",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleExecution",
    "clientId": "iotconsole-123456789012-3",
    "topicName": "test-topic",
    "ruleName": "rule1",
    "ruleAction": "DynamoAction",
    "resources": {
        "ItemHashKeyField": "id",
        "Table": "trashbin",
        "Operation": "Insert",
        "ItemHashKeyValue": "id",
        "IsPayloadJSON": "true"
    },
    "principalId": "ABCDEFG1234567ABCD890:outis",
    "details": "User: arn:aws:sts::123456789012:assumed-role/dynamo-testbin/5aUMInJH
 is not authorized to perform: dynamodb:PutItem on resource: arn:aws:dynamodb:us-
east-1:123456789012:table/testbin (Service: AmazonDynamoDBv2; Status Code: 400; Error Code:
 AccessDeniedException; Request ID: AKQJ987654321AKQJ123456789AKQJ987654321AKQJ987654321)"
}
```

Here is a similar example log generated by global logging (p. 344):

```
2017-12-09 22:49:17.954 TRACEID:ff562535-6964-506a-e141-78d40375fc4e
PRINCIPALID:ABCDEFG1234567ABCD890:outis [ERROR] EVENT:DynamoActionFailure
TOPICNAME:test-topic CLIENTID:iotconsole-123456789012-3
MESSAGE:Dynamo Insert record failed. The error received was User:
 arn:aws:sts::123456789012:assumed-role/dynamo-testbin/5aUMInJI is not authorized to
 perform: dynamodb:PutItem on resource: arn:aws:dynamodb:us-east-1:123456789012:table/
testbin
```

```
(Service: AmazonDynamoDBv2; Status Code: 400; Error Code: AccessDeniedException; Request
 ID: AKQJ987654321AKQJ987654321AKQJ987654321AKQJ987654321).
Message arrived on: test-topic, Action: dynamo, Table: trashbin, HashKeyField: id,
 HashKeyValue: id, RangeKeyField: None, RangeKeyValue: 123456789012
No newer events found at the moment. Retry.
```

For more information, see the section called “Viewing Logs” (p. 358).

External services are controlled by the end user. Before rule execution, make sure external services are set up with enough throughput and capacity units.

Diagnosing Problems with Thing Shadows

Diagnosing Thing Shadows

Issue	Troubleshooting Guidelines
A thing shadow document is rejected with "Invalid JSON document."	If you are unfamiliar with JSON, modify the examples provided in this guide for your own use. For more information, see Thing Shadow Document Syntax.
I submitted correct JSON, but none or only parts of it are stored in the thing shadow document.	Be sure you are following the JSON formatting guidelines. Only JSON fields in the `desired` and `reported` sections are stored. JSON content (even if formally correct) outside of those sections is ignored.
I received an error that the thing shadow exceeds the allowed size.	The thing shadow supports 8 KB of data only. Try shortening field names inside of your JSON document or simply create more thing shadows. A device can have an unlimited number of thing shadows. The only requirement is that the thing name is unique in your account.
When I receive a thing shadow, it is larger than 8 KB. How can this happen?	Upon receipt, the AWS IoT service adds metadata to the thing shadow. The service includes this data in its response, but it does not count toward the limit of 8 KB. Only the data for `desired` and `reported` state inside the state document sent to the thing shadow counts toward the limit.
My request has been rejected due to incorrect version. What should I do?	Perform a GET operation to sync to the latest state document version. When using MQTT, subscribe to the ./update/accepted topic to be notified about state changes and receive the latest version of the JSON document.
The timestamp is off by several seconds.	The timestamp for individual fields and the whole JSON document is updated when the document is received by the AWS IoT service or when the state document is published onto the ./update/accepted and ./update/delta message. Messages can be delayed over the network, which can cause the timestamp to be off by a few seconds.

Issue	Troubleshooting Guidelines
My device can publish and subscribe on the corresponding thing shadow topics, but when I attempt to update the thing shadow document over the HTTP REST API, I get HTTP 403.	Be sure you have created policies in IAM to allow access to these topics and for the corresponding action (UPDATE/GET/DELETE) for the credentials you are using. IAM policies and certificate policies are independent.
Other issues.	The Thing Shadows service logs errors to CloudWatch Logs. To identify device and configuration issues, enable CloudWatch Logs and view the logs for debug information.

Diagnosing Salesforce IoT Input Stream Action Issues

Execution Trace

How do I see the execution trace of a Salesforce action?

If CloudWatch Logs are not set up, see the Setting Up CloudWatch Logs with AWS IoT (p. 342) section. After you have activated the logs, you are able to see the execution trace of the Salesforce action.

Action Success and Failure

How do I check that messages have been sent successfully to a Salesforce IoT input stream?

View the logs generated by execution of the Salesforce action in CloudWatch Logs. If you see "`Action executed successfully`," then it means that the AWS IoT rules engine received confirmation from the Salesforce IoT that the message was successfully pushed to the targeted input stream.

If you are experiencing problems with the Salesforce IoT platform, contact Salesforce IoT support.

What do I do if messages have not been sent successfully to a Salesforce IoT input stream?

View the logs generated by execution of the Salesforce action in CloudWatch Logs. Depending on the log entry, you can try the following actions:

`Failed to locate the host`

Check that the `url` parameter of the action is correct and that your Salesforce IoT input stream exists.

`Received Internal Server Error from Salesforce`

Retry. If the problem persists, contact Salesforce IoT Support.

`Received Bad Request Exception from Salesforce`

Check the payload you are sending for errors.

`Received Unsupported Media Type Exception from Salesforce`

Salesforce IoT does not support a binary payload at this time. Check that you are sending a JSON payload.

`Received Unauthorized Exception from Salesforce`

Check that the `token` parameter of the action is correct and that your token is still valid.

`Received Not Found Exception from Salesforce`

Check that the `url` parameter of the action is correct and that your Salesforce IoT input stream exists.

If you receive an error that is not listed here, contact AWS Support.

AWS IoT Errors

This section lists the error codes sent by AWS IoT.

Message Broker Error Codes

Error Code	Error Description
400	Bad request.
401	Unauthorized.
403	Forbidden.
503	Service unavailable.

Identity and Security Error Codes

Error Code	Error Description
401	Unauthorized.

Thing Shadow Error Codes

Error Code	Error Description
400	Bad request.
401	Unauthorized.
403	Forbidden.
404	Not found.
409	Conflict.
413	Request too large.
422	Failed to process request.
429	Too many requests.
500	Internal error.
503	Service unavailable.

IoT Commands

This chapter contains the following sections:

- UpdateEventConfigurations (p. 727)
- UpdateIndexingConfiguration (p. 729)
- UpdateJobExecution (p. 731)
- UpdateRoleAlias (p. 736)
- UpdateStream (p. 739)
- UpdateThing (p. 742)
- UpdateThingGroup (p. 746)
- UpdateThingGroupsForThing (p. 749)
- UpdateThingShadow (p. 751)

AcceptCertificateTransfer

Accepts a pending certificate transfer. The default state of the certificate is INACTIVE.

To check for pending certificate transfers, call ListCertificates to enumerate your certificates.

Request syntax:

```
PATCH /accept-certificate-transfer/certificateId?setAsActive=setAsActive
```

URI Request Parameters:

Name	Type	Req?	Description
certificateId	CertificateId	yes	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)
setAsActive	SetAsActive	no	Specifies whether the certificate is active.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`TransferAlreadyCompletedException`

You can't revert the certificate transfer because the transfer is already complete.

HTTP response code: 410

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  accept-certificate-transfer \
    --certificate-id <value> \
    [--set-as-active | --no-set-as-active]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "certificateId": "string",
  "setAsActive": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)
setAsActive	boolean	Specifies whether the certificate is active.

Output:

None

AddThingToThingGroup

Adds a thing to a thing group.

Request syntax:

```
PUT /thing-groups/addThingToThingGroup
Content-type: application/json

{
  "thingGroupName": "string",
  "thingGroupArn": "string",
  "thingName": "string",
  "thingArn": "string"
}
```

Request Body Parameters:

Name	Type	Req?	Description
thingGroupName	ThingGroupName	no	The name of the group to which you are adding a thing.
thingGroupArn	ThingGroupArn	no	The ARN of the group to which you are adding a thing.
thingName	ThingName	no	The name of the thing to add to a group.
thingArn	ThingArn	no	The ARN of the thing to add to a group.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  add-thing-to-thing-group \
    [--thing-group-name <value>] \
    [--thing-group-arn <value>] \
    [--thing-name <value>] \
    [--thing-arn <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingGroupName": "string",
  "thingGroupArn": "string",
  "thingName": "string",
  "thingArn": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
thingGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the group to which you are adding a thing.
thingGroupArn	string	The ARN of the group to which you are adding a thing.
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing to add to a group.
thingArn	string	The ARN of the thing to add to a group.

Output:

None

AssociateTargetsWithJob

Associates a group with a continuous job. The following criteria must be met:

- The job must have been created with the `targetSelection` field set to "CONTINUOUS".
- The job status must currently be "IN_PROGRESS".

- The total number of targets associated with a job must not exceed 100.

Request syntax:

```
POST /jobs/jobId/targets
Content-type: application/json

{
  "targets": [
    "string"
  ],
  "comment": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
jobId	JobId	yes	The unique identifier you assigned to this job when it was created.

Request Body Parameters:

Name	Type	Req?	Description
targets	JobTargets	yes	A list of thing group ARNs that define the targets of the job.
comment	Comment	no	An optional comment string describing why the job was associated with the targets.

Response syntax:

```
Content-type: application/json

{
  "jobArn": "string",
  "jobId": "string",
  "description": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
jobArn	JobArn	no	An ARN identifying the job.
jobId	JobId	no	The unique identifier you assigned to this job when it was created.

Name	Type	Req?	Description
description	JobDescription	no	A short text description of the job.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`LimitExceededException`

The number of attached entities exceeds the limit.

HTTP response code: 410

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  associate-targets-with-job \
    --targets <value> \
    --job-id <value> \
    [--comment <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "targets": [
    "string"
  ],
  "jobId": "string",
  "comment": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
targets	list member: TargetArn	A list of thing group ARNs that define the targets of the job.
TargetArn	string	
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.
comment	string length max:2028 pattern: [^\\p{C}]+	An optional comment string describing why the job was associated with the targets.

Output:

```
{
  "jobArn": "string",
  "jobId": "string",
  "description": "string"
}
```

cli output fields:

Name	Type	Description
jobArn	string	An ARN identifying the job.
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.
description	string length max:2028 pattern: [^\\p{C}]+	A short text description of the job.

AttachPolicy

Attaches a policy to the specified target.

Request syntax:

```
PUT /target-policies/policyName
Content-type: application/json
```

```
{
  "target": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
policyName	PolicyName	yes	The name of the policy to attach.

Request Body Parameters:

Name	Type	Req?	Description
target	PolicyTarget	yes	The identity to which the policy is attached.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`LimitExceededException`

The number of attached entities exceeds the limit.

HTTP response code: 410

CLI

Synopsis:

```
aws iot  attach-policy \
    --policy-name <value> \
    --target <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "policyName": "string",
  "target": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The name of the policy to attach.
target	string	The identity to which the policy is attached.

Output:

None

AttachPrincipalPolicy

Attaches the specified policy to the specified principal (certificate or other credential).

Note: This API is deprecated. Please use AttachPolicy instead.

Request syntax:

```
PUT /principal-policies/policyName
x-amzn-iot-principal: principal
```

URI Request Parameters:

Name	Type	Req?	Description
policyName	PolicyName	yes	The policy name.

Name	Type	Req?	Description
principal	Principal	yes	The principal, which can be a certificate ARN (as returned from the CreateCertificate operation) or an Amazon Cognito ID.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`LimitExceededException`

The number of attached entities exceeds the limit.

HTTP response code: 410

CLI

Synopsis:

```
aws iot  attach-principal-policy \
    --policy-name <value> \
    --principal <value>  \
```

```
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "policyName": "string",
  "principal": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.
principal	string	The principal, which can be a certificate ARN (as returned from the CreateCertificate operation) or an Amazon Cognito ID.

Output:

None

AttachThingPrincipal

Attaches the specified principal to the specified thing.

Request syntax:

```
PUT /things/thingName/principals
x-amzn-principal: principal
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	yes	The name of the thing.
principal	Principal	yes	The principal, such as a certificate or other credential.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  attach-thing-principal \
    --thing-name <value> \
    --principal <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string",
  "principal": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing.

Name	Type	Description
principal	string	The principal, such as a certificate or other credential.

Output:

None

CancelCertificateTransfer

Cancels a pending transfer for the specified certificate.

Note Only the transfer source account can use this operation to cancel a transfer. (Transfer destinations can use RejectCertificateTransfer instead.) After transfer, AWS IoT returns the certificate to the source account in the INACTIVE state. After the destination account has accepted the transfer, the transfer cannot be cancelled.

After a certificate transfer is cancelled, the status of the certificate changes from PENDING_TRANSFER to INACTIVE.

Request syntax:

```
PATCH /cancel-certificate-transfer/certificateId
```

URI Request Parameters:

Name	Type	Req?	Description
certificateId	CertificateId	yes	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`TransferAlreadyCompletedException`

You can't revert the certificate transfer because the transfer is already complete.

HTTP response code: 410

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  cancel-certificate-transfer \
    --certificate-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "certificateId": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)

Output:

None

CancelJob

Cancels a job.

Request syntax:

```
PUT /jobs/jobId/cancel
Content-type: application/json

{
  "comment": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
jobId	JobId	yes	The unique identifier you assigned to this job when it was created.

Request Body Parameters:

Name	Type	Req?	Description
comment	Comment	no	An optional comment string describing why the job was canceled.

Response syntax:

```
Content-type: application/json

{
  "jobArn": "string",
  "jobId": "string",
  "description": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
jobArn	JobArn	no	The job ARN.
jobId	JobId	no	The unique identifier you assigned to this job when it was created.
description	JobDescription	no	A short text description of the job.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  cancel-job \
    --job-id <value> \
    [--comment <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "jobId": "string",
  "comment": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.
comment	string length max:2028 pattern: [^\\p{C}]+	An optional comment string describing why the job was canceled.

Output:

```
{
  "jobArn": "string",
  "jobId": "string",
  "description": "string"
```

```
}
```

cli output fields:

Name	Type	Description
jobArn	string	The job ARN.
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.
description	string length max:2028 pattern: [^\\p{C}]+	A short text description of the job.

ClearDefaultAuthorizer

Clears the default authorizer.

Request syntax:

```
DELETE /default-authorizer
```

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  clear-default-authorizer  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
}
```

Output:

None

CreateAuthorizer

Creates an authorizer.

Request syntax:

```
POST /authorizer/authorizerName
Content-type: application/json

{
  "authorizerFunctionArn": "string",
  "tokenKeyName": "string",
  "tokenSigningPublicKeys": {
    "string": "string"
  },
  "status": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
authorizerName	AuthorizerName	yes	The authorizer name.

Request Body Parameters:

Name	Type	Req?	Description
authorizerFunctionArn	AuthorizerFunctionArn	yes	The ARN of the authorizer's Lambda function.

Name	Type	Req?	Description
tokenKeyName	TokenKeyName	yes	The name of the token key used to extract the token from the HTTP headers.
tokenSigningPublicKeys	PublicKeyMap	yes	The public keys used to verify the digital signature returned by your custom authentication service.
status	AuthorizerStatus	no	The status of the create authorizer request.

Response syntax:

```
Content-type: application/json

{
  "authorizerName": "string",
  "authorizerArn": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
authorizerName	AuthorizerName	no	The authorizer's name.
authorizerArn	AuthorizerArn	no	The authorizer ARN.

Errors:

`ResourceAlreadyExistsException`

The resource already exists.

HTTP response code: 409

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`LimitExceededException`

The number of attached entities exceeds the limit.

HTTP response code: 410

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  create-authorizer \
    --authorizer-name <value> \
    --authorizer-function-arn <value> \
    --token-key-name <value> \
    --token-signing-public-keys <value> \
    [--status <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "authorizerName": "string",
  "authorizerFunctionArn": "string",
  "tokenKeyName": "string",
  "tokenSigningPublicKeys": {
    "string": "string"
  },
  "status": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
authorizerName	string length max:128 min:1 pattern: [w=,@-]+	The authorizer name.
authorizerFunctionArn	string	The ARN of the authorizer's Lambda function.
tokenKeyName	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The name of the token key used to extract the token from the HTTP headers.

Name	Type	Description
tokenSigningPublicKeys	map key: KeyName value: KeyValue	The public keys used to verify the digital signature returned by your custom authentication service.
KeyName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
KeyValue	string length max:5120	
status	string enum: ACTIVE | INACTIVE java class: iot.identity.service.AuthorizerStatus	The status of the create authorizer request.

Output:

```
{
  "authorizerName": "string",
  "authorizerArn": "string"
}
```

cli output fields:

Name	Type	Description
authorizerName	string length max:128 min:1 pattern: [w=,@-]+	The authorizer's name.
authorizerArn	string	The authorizer ARN.

CreateCertificateFromCsr

Creates an X.509 certificate using the specified certificate signing request.

Note: The CSR must include a public key that is either an RSA key with a length of at least 2048 bits or an ECC key from NIST P-256 or NIST P-384 curves.

Note: Reusing the same certificate signing request (CSR) results in a distinct certificate.

You can create multiple certificates in a batch by creating a directory, copying multiple .csr files into that directory, and then specifying that directory on the command line. The following commands show how to create a batch of certificates given a batch of CSRs.

Assuming a set of CSRs are located inside of the directory my-csr-directory:

On Linux and OS X, the command is:

$ ls my-csr-directory/ | xargs -I aws iot create-certificate-from-csr --certificate-signing-request file://my-csr-directory/

This command lists all of the CSRs in my-csr-directory and pipes each CSR file name to the aws iot create-certificate-from-csr AWS CLI command to create a certificate for the corresponding CSR.

The aws iot create-certificate-from-csr part of the command can also be run in parallel to speed up the certificate creation process:

$ ls my-csr-directory/ | xargs -P 10 -I aws iot create-certificate-from-csr --certificate-signing-request file://my-csr-directory/

On Windows PowerShell, the command to create certificates for all CSRs in my-csr-directory is:

> ls -Name my-csr-directory | % *aws iot create-certificate-from-csr --certificate-signing-request file://my-csr-directory/$_*

On a Windows command prompt, the command to create certificates for all CSRs in my-csr-directory is:

> forfiles /p my-csr-directory /c "cmd /c aws iot create-certificate-from-csr --certificate-signing-request file://@path"

Request syntax:

```
POST /certificates?setAsActive=setAsActive
Content-type: application/json

{
  "certificateSigningRequest": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
setAsActive	SetAsActive	no	Specifies whether the certificate is active.

Request Body Parameters:

Name	Type	Req?	Description
certificateSigningRequest	CertificateSigningRequest	yes	The certificate signing request (CSR).

Response syntax:

```
Content-type: application/json

{
  "certificateArn": "string",
  "certificateId": "string",
```

```
  "certificatePem": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
certificateArn	CertificateArn	no	The Amazon Resource Name (ARN) of the certificate. You can use the ARN as a principal for policy operations.
certificateId	CertificateId	no	The ID of the certificate. Certificate management operations only take a certificateId.
certificatePem	CertificatePem	no	The certificate data, in PEM format.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  create-certificate-from-csr \
    --certificate-signing-request <value> \
    [--set-as-active | --no-set-as-active]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "certificateSigningRequest": "string",
  "setAsActive": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
certificateSigningRequest	string length min:1	The certificate signing request (CSR).
setAsActive	boolean	Specifies whether the certificate is active.

Output:

```
{
  "certificateArn": "string",
  "certificateId": "string",
  "certificatePem": "string"
}
```

cli output fields:

Name	Type	Description
certificateArn	string	The Amazon Resource Name (ARN) of the certificate. You can use the ARN as a principal for policy operations.
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the certificate. Certificate management operations only take a certificateId.
certificatePem	string length max:65536 min:1	The certificate data, in PEM format.

CreateJob

Creates a job.

Request syntax:

```
PUT /jobs/jobId
Content-type: application/json

{
  "targets": [
    "string"
  ],
  "documentSource": "string",
  "document": "string",
  "description": "string",
  "presignedUrlConfig": {
    "roleArn": "string",
    "expiresInSec": "long"
  },
  "targetSelection": "string",
  "jobExecutionsRolloutConfig": {
    "maximumPerMinute": "integer"
  },
  "documentParameters": {
    "string": "string"
  }
}
```

URI Request Parameters:

Name	Type	Req?	Description
jobId	JobId	yes	A job identifier which must be unique for your AWS account. We recommend using a UUID. Alpha-numeric characters, "-" and "_" are valid for use here.

Request Body Parameters:

Name	Type	Req?	Description
targets	JobTargets	yes	A list of things and thing groups to which the job should be sent.
documentSource	JobDocumentSource	no	An S3 link to the job document.
document	JobDocument	no	The job document.
description	JobDescription	no	A short text description of the job.
presignedUrlConfig	PresignedUrlConfig	no	Configuration information for pre-signed S3 URLs.
targetSelection	TargetSelection	no	Specifies whether the job will continue to run (CONTINUOUS),

Name	Type	Req?	Description
			or will be complete after all those things specified as targets have completed the job (SNAPSHOT). If continuous, the job may also be run on a thing when a change is detected in a target. For example, a job will run on a thing when the thing is added to a target group, even after the job was completed by all things originally in the group.
jobExecutionsRolloutConfig	JobExecutionsRolloutConfig	no	Allows you to create a staged rollout of the job.
documentParameters	JobDocumentParameters	no	Parameters for the job document.

Response syntax:

```
Content-type: application/json

{
  "jobArn": "string",
  "jobId": "string",
  "description": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
jobArn	JobArn	no	The job ARN.
jobId	JobId	no	The unique identifier you assigned to this job.
description	JobDescription	no	The job description.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ResourceAlreadyExistsException`

The resource already exists.

HTTP response code: 409

`LimitExceededException`

The number of attached entities exceeds the limit.

HTTP response code: 410

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  create-job \
    --job-id <value> \
    --targets <value> \
    [--document-source <value>] \
    [--document <value>] \
    [--description <value>] \
    [--presigned-url-config <value>] \
    [--target-selection <value>] \
    [--job-executions-rollout-config <value>] \
    [--document-parameters <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "jobId": "string",
  "targets": [
    "string"
  ],
  "documentSource": "string",
  "document": "string",
  "description": "string",
  "presignedUrlConfig": {
    "roleArn": "string",
    "expiresInSec": "long"
  },
  "targetSelection": "string",
  "jobExecutionsRolloutConfig": {
    "maximumPerMinute": "integer"
  },
```

```
  "documentParameters": {
    "string": "string"
  }
}
```

`cli-input-json` fields:

Name	Type	Description
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	A job identifier which must be unique for your AWS account. We recommend using a UUID. Alpha-numeric characters, "-" and "_" are valid for use here.
targets	list member: TargetArn	A list of things and thing groups to which the job should be sent.
TargetArn	string	
documentSource	string length max:1350 min:1	An S3 link to the job document.
document	string length max:32768	The job document.
description	string length max:2028 pattern: [^\\p{C}]+	A short text description of the job.
presignedUrlConfig	PresignedUrlConfig	Configuration information for pre-signed S3 URLs.
roleArn	string length max:2048 min:20	The ARN of an IAM role that grants grants permission to download files from the S3 bucket where the job data/ updates are stored. The role must also grant permission for IoT to download the files.
expiresInSec	long java class: java.lang.Long range- max:3600 min:60	How long (in seconds) pre-signed URLs are valid. Valid values are 60 - 3600, the default value is 3600 seconds. Pre-signed URLs are generated when Jobs receives an MQTT request for the job document.
targetSelection	string enum: CONTINUOUS | SNAPSHOT java class: com.amazonaws.iot.laser.TargetSelection	Specifies whether the job will continue to run (CONTINUOUS), or will be complete after all those things specified as targets have completed the job (SNAPSHOT). If continuous, the

Name	Type	Description
		job may also be run on a thing when a change is detected in a target. For example, a job will run on a thing when the thing is added to a target group, even after the job was completed by all things originally in the group.
jobExecutionsRolloutConfig	JobExecutionsRolloutConfig	Allows you to create a staged rollout of the job.
maximumPerMinute	integer java class: java.lang.Integer range- max:1000 min:1	The maximum number of things that will be notified of a pending job, per minute. This parameter allows you to create a staged rollout.
documentParameters	map key: ParameterKey value: ParameterValue	Parameters for the job document.
ParameterKey	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
ParameterValue	string length max:1024 min:1 pattern: [^\\p{C}]+	

Output:

```
{
  "jobArn": "string",
  "jobId": "string",
  "description": "string"
}
```

cli output fields:

Name	Type	Description
jobArn	string	The job ARN.
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job.
description	string length max:2028	The job description.

Name	Type	Description
	pattern: [^\\p{C}]+	

CreateKeysAndCertificate

Creates a 2048-bit RSA key pair and issues an X.509 certificate using the issued public key.

Note This is the only time AWS IoT issues the private key for this certificate, so it is important to keep it in a secure location.

Request syntax:

```
POST /keys-and-certificate?setAsActive=setAsActive
```

URI Request Parameters:

Name	Type	Req?	Description
setAsActive	SetAsActive	no	Specifies whether the certificate is active.

Response syntax:

```
Content-type: application/json

{
  "certificateArn": "string",
  "certificateId": "string",
  "certificatePem": "string",
  "keyPair": {
    "PublicKey": "string",
    "PrivateKey": "string"
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
certificateArn	CertificateArn	no	The ARN of the certificate.
certificateId	CertificateId	no	The ID of the certificate. AWS IoT issues a default subject name for the certificate (for example, AWS IoT Certificate).
certificatePem	CertificatePem	no	The certificate data, in PEM format.
keyPair	KeyPair	no	The generated key pair.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  create-keys-and-certificate \
    [--set-as-active | --no-set-as-active]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "setAsActive": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
setAsActive	boolean	Specifies whether the certificate is active.

Output:

```
{
  "certificateArn": "string",
  "certificateId": "string",
  "certificatePem": "string",
  "keyPair": {
    "PublicKey": "string",
    "PrivateKey": "string"
  }
}
```

cli output fields:

Name	Type	Description
certificateArn	string	The ARN of the certificate.
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the certificate. AWS IoT issues a default subject name for the certificate (for example, AWS IoT Certificate).
certificatePem	string length max:65536 min:1	The certificate data, in PEM format.
keyPair	KeyPair	The generated key pair.
PublicKey	string length min:1	The public key.
PrivateKey	string length min:1	The private key.

CreateOTAUpdate

Creates an AWS IoT OTAUpdate on a target group of things or groups.

Request syntax:

```
POST /otaUpdates/otaUpdateId
Content-type: application/json

{
  "description": "string",
  "targets": [
    "string"
  ],
  "targetSelection": "string",
  "files": [
    {
      "fileName": "string",
      "fileVersion": "string",
      "fileSource": {
        "streamId": "string",
        "fileId": "integer"
      },
```

```
      "codeSigning": {
        "awsSignerJobId": "string",
        "customCodeSigning": {
          "signature": {
            "stream": {
              "streamId": "string",
              "fileId": "integer"
            },
            "inlineDocument": "blob"
          },
          "certificateChain": {
            "stream": {
              "streamId": "string",
              "fileId": "integer"
            },
            "certificateName": "string",
            "inlineDocument": "string"
          },
          "hashAlgorithm": "string",
          "signatureAlgorithm": "string"
        }
      },
      "attributes": {
        "string": "string"
      }
    }
  ],
  "roleArn": "string",
  "additionalParameters": {
    "string": "string"
  }
}
```

URI Request Parameters:

Name	Type	Req?	Description
otaUpdateId	OTAUpdateId	yes	The ID of the OTA update to be created.

Request Body Parameters:

Name	Type	Req?	Description
description	OTAUpdateDescription	no	The description of the OTA update.
targets	Targets	yes	The targeted devices to receive OTA updates.
targetSelection	TargetSelection	no	Specifies whether the update will continue to run (CONTINUOUS), or will be complete after all the things specified as targets have completed the update (SNAPSHOT). If continuous, the update may also be run on a thing when a

Name	Type	Req?	Description
			change is detected in a target. For example, an update will run on a thing when the thing is added to a target group, even after the update was completed by all things originally in the group. Valid values: CONTINUOUS | SNAPSHOT.
files	OTAUpdateFiles	yes	The files to be streamed by the OTA update.
roleArn	RoleArn	yes	The IAM role that allows access to the AWS IoT Jobs service.
additionalParameters	AdditionalParameterMap	no	A list of additional OTA update parameters which are name-value pairs.

Response syntax:

```
Content-type: application/json

{
  "otaUpdateId": "string",
  "awsIotJobId": "string",
  "otaUpdateArn": "string",
  "awsIotJobArn": "string",
  "otaUpdateStatus": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
otaUpdateId	OTAUpdateId	no	The OTA update ID.
awsIotJobId	AwsIotJobId	no	The AWS IoT job ID associated with the OTA update.
otaUpdateArn	OTAUpdateArn	no	The OTA update ARN.
awsIotJobArn	AwsIotJobArn	no	The AWS IoT job ARN associated with the OTA update.
otaUpdateStatus	OTAUpdateStatus	no	The OTA update status.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ResourceAlreadyExistsException`

The resource already exists.

HTTP response code: 409

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  create-ota-update \
    --ota-update-id <value> \
    [--description <value>] \
    --targets <value> \
    [--target-selection <value>] \
    --files <value> \
    --role-arn <value> \
    [--additional-parameters <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "otaUpdateId": "string",
  "description": "string",
  "targets": [
    "string"
```

```
  ],
  "targetSelection": "string",
  "files": [
    {
      "fileName": "string",
      "fileVersion": "string",
      "fileSource": {
        "streamId": "string",
        "fileId": "integer"
      },
      "codeSigning": {
        "awsSignerJobId": "string",
        "customCodeSigning": {
          "signature": {
            "stream": {
              "streamId": "string",
              "fileId": "integer"
            },
            "inlineDocument": "blob"
          },
          "certificateChain": {
            "stream": {
              "streamId": "string",
              "fileId": "integer"
            },
            "certificateName": "string",
            "inlineDocument": "string"
          },
          "hashAlgorithm": "string",
          "signatureAlgorithm": "string"
        }
      },
      "attributes": {
        "string": "string"
      }
    }
  ],
  "roleArn": "string",
  "additionalParameters": {
    "string": "string"
  }
}
```

`cli-input-json` fields:

Name	Type	Description
otaUpdateId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The ID of the OTA update to be created.
description	string length max:2028 pattern: [^\\p{C}]+	The description of the OTA update.
targets	list member: Target	The targeted devices to receive OTA updates.
Target	string	

Name	Type	Description
targetSelection	string enum: CONTINUOUS | SNAPSHOT	Specifies whether the update will continue to run (CONTINUOUS), or will be complete after all the things specified as targets have completed the update (SNAPSHOT). If continuous, the update may also be run on a thing when a change is detected in a target. For example, an update will run on a thing when the thing is added to a target group, even after the update was completed by all things originally in the group. Valid values: CONTINUOUS | SNAPSHOT.
files	list member: OTAUpdateFile	The files to be streamed by the OTA update.
OTAUpdateFile	OTAUpdateFile	
fileName	string	The name of the file.
fileVersion	string	The file version.
fileSource	Stream	The source of the file.
streamId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The stream ID.
fileId	integer java class: java.lang.Integer range- max:255 min:0	The ID of a file associated with a stream.
codeSigning	CodeSigning	The code signing method of the file.
awsSignerJobId	string	The ID of the AWSSignerJob which was created to sign the file.
customCodeSigning	CustomCodeSigning	A custom method for code signing a file.
signature	CodeSigningSignature	The signature for the file.
stream	Stream	A stream of the code signing signature.

Name	Type	Description
streamId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The stream ID.
fileId	integer java class: java.lang.Integer range- max:255 min:0	The ID of a file associated with a stream.
inlineDocument	blob	A base64 encoded binary representation of the code signing signature.
certificateChain	CodeSigningCertificateChain	The certificate chain.
stream	Stream	A stream of the certificate chain files.
streamId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The stream ID.
fileId	integer java class: java.lang.Integer range- max:255 min:0	The ID of a file associated with a stream.
certificateName	string	The name of the certificate.
inlineDocument	string	A base64 encoded binary representation of the code signing certificate chain.
hashAlgorithm	string	The hash algorithm used to code sign the file.
signatureAlgorithm	string	The signature algorithm used to code sign the file.
attributes	map key: Key value: Value	A list of name/attribute pairs.
Key	string	
Value	string	
roleArn	string length max:2048 min:20	The IAM role that allows access to the AWS IoT Jobs service.

Name	Type	Description
additionalParameters	map key: Key value: Value	A list of additional OTA update parameters which are name-value pairs.
Key	string	
Value	string	

Output:

```
{
  "otaUpdateId": "string",
  "awsIotJobId": "string",
  "otaUpdateArn": "string",
  "awsIotJobArn": "string",
  "otaUpdateStatus": "string"
}
```

cli output fields:

Name	Type	Description
otaUpdateId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The OTA update ID.
awsIotJobId	string	The AWS IoT job ID associated with the OTA update.
otaUpdateArn	string	The OTA update ARN.
awsIotJobArn	string	The AWS IoT job ARN associated with the OTA update.
otaUpdateStatus	string enum: CREATE_PENDING | CREATE_IN_PROGRESS | CREATE_COMPLETE | CREATE_FAILED	The OTA update status.

CreatePolicy

Creates an AWS IoT policy.

The created policy is the default version for the policy. This operation creates a policy version with a version identifier of **1** and sets **1** as the policy's default version.

Request syntax:

```
POST /policies/policyName
```

```
Content-type: application/json

{
  "policyDocument": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
policyName	PolicyName	yes	The policy name.

Request Body Parameters:

Name	Type	Req?	Description
policyDocument	PolicyDocument	yes	The JSON document that describes the policy. **policyDocument** must have a minimum length of 1, with a maximum length of 2048, excluding whitespace.

Response syntax:

```
Content-type: application/json

{
  "policyName": "string",
  "policyArn": "string",
  "policyDocument": "string",
  "policyVersionId": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
policyName	PolicyName	no	The policy name.
policyArn	PolicyArn	no	The policy ARN.
policyDocument	PolicyDocument	no	The JSON document that describes the policy.
policyVersionId	PolicyVersionId	no	The policy version ID.

Errors:

`ResourceAlreadyExistsException`

The resource already exists.

HTTP response code: 409

`MalformedPolicyException`

The policy documentation is not valid.

HTTP response code: 400

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  create-policy \
    --policy-name <value> \
    --policy-document <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "policyName": "string",
  "policyDocument": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
policyName	string	The policy name.

Name	Type	Description
	length max:128 min:1 pattern: [w+=,.@-]+	
policyDocument	string	The JSON document that describes the policy. **policyDocument** must have a minimum length of 1, with a maximum length of 2048, excluding whitespace.

Output:

```
{
  "policyName": "string",
  "policyArn": "string",
  "policyDocument": "string",
  "policyVersionId": "string"
}
```

cli output fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.
policyArn	string	The policy ARN.
policyDocument	string	The JSON document that describes the policy.
policyVersionId	string pattern: [0-9]+	The policy version ID.

CreatePolicyVersion

Creates a new version of the specified AWS IoT policy. To update a policy, create a new policy version. A managed policy can have up to five versions. If the policy has five versions, you must use DeletePolicyVersion to delete an existing version before you create a new one.

Optionally, you can set the new version as the policy's default version. The default version is the operative version (that is, the version that is in effect for the certificates to which the policy is attached).

Request syntax:

```
POST /policies/policyName/version?setAsDefault=setAsDefault
Content-type: application/json
```

```
{
  "policyDocument": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
policyName	PolicyName	yes	The policy name.
setAsDefault	SetAsDefault	no	Specifies whether the policy version is set as the default. When this parameter is true, the new policy version becomes the operative version (that is, the version that is in effect for the certificates to which the policy is attached).

Request Body Parameters:

Name	Type	Req?	Description
policyDocument	PolicyDocument	yes	The JSON document that describes the policy. Minimum length of 1. Maximum length of 2048, excluding whitespace.

Response syntax:

```
Content-type: application/json

{
  "policyArn": "string",
  "policyDocument": "string",
  "policyVersionId": "string",
  "isDefaultVersion": "boolean"
}
```

Response Body Parameters:

Name	Type	Req?	Description
policyArn	PolicyArn	no	The policy ARN.
policyDocument	PolicyDocument	no	The JSON document that describes the policy.
policyVersionId	PolicyVersionId	no	The policy version ID.

Name	Type	Req?	Description
isDefaultVersion	IsDefaultVersion	no	Specifies whether the policy version is the default.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`MalformedPolicyException`

The policy documentation is not valid.

HTTP response code: 400

`VersionsLimitExceededException`

The number of policy versions exceeds the limit.

HTTP response code: 409

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  create-policy-version \
    --policy-name <value> \
```

```
    --policy-document <value> \
    [--set-as-default | --no-set-as-default]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "policyName": "string",
  "policyDocument": "string",
  "setAsDefault": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.
policyDocument	string	The JSON document that describes the policy. Minimum length of 1. Maximum length of 2048, excluding whitespace.
setAsDefault	boolean	Specifies whether the policy version is set as the default. When this parameter is true, the new policy version becomes the operative version (that is, the version that is in effect for the certificates to which the policy is attached).

Output:

```
{
  "policyArn": "string",
  "policyDocument": "string",
  "policyVersionId": "string",
  "isDefaultVersion": "boolean"
}
```

cli output fields:

Name	Type	Description
policyArn	string	The policy ARN.
policyDocument	string	The JSON document that describes the policy.
policyVersionId	string pattern: [0-9]+	The policy version ID.

Name	Type	Description
isDefaultVersion	boolean	Specifies whether the policy version is the default.

CreateRoleAlias

Creates a role alias.

Request syntax:

```
POST /role-aliases/roleAlias
Content-type: application/json

{
  "roleArn": "string",
  "credentialDurationSeconds": "integer"
}
```

URI Request Parameters:

Name	Type	Req?	Description
roleAlias	RoleAlias	yes	The role alias that points to a role ARN. This allows you to change the role without having to update the device.

Request Body Parameters:

Name	Type	Req?	Description
roleArn	RoleArn	yes	The role ARN.
credentialDurationSeconds	CredentialDurationSeconds	no	How long (in seconds) the credentials will be valid.

Response syntax:

```
Content-type: application/json

{
  "roleAlias": "string",
  "roleAliasArn": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
roleAlias	RoleAlias	no	The role alias.

Name	Type	Req?	Description
roleAliasArn	RoleAliasArn	no	The role alias ARN.

Errors:

`ResourceAlreadyExistsException`

The resource already exists.

HTTP response code: 409

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`LimitExceededException`

The number of attached entities exceeds the limit.

HTTP response code: 410

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  create-role-alias \
    --role-alias <value> \
    --role-arn <value> \
    [--credential-duration-seconds <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "roleAlias": "string",
  "roleArn": "string",
  "credentialDurationSeconds": "integer"
}
```

`cli-input-json` fields:

Name	Type	Description
roleAlias	string length max:128 min:1 pattern: [w=,@-]+	The role alias that points to a role ARN. This allows you to change the role without having to update the device.
roleArn	string length max:2048 min:20	The role ARN.
credentialDurationSeconds	integer java class: java.lang.Integer range- max:3600 min:900	How long (in seconds) the credentials will be valid.

Output:

```
{
  "roleAlias": "string",
  "roleAliasArn": "string"
}
```

cli output fields:

Name	Type	Description
roleAlias	string length max:128 min:1 pattern: [w=,@-]+	The role alias.
roleAliasArn	string	The role alias ARN.

CreateStream

Creates a stream for delivering one or more large files in chunks over MQTT. A stream transports data bytes in chunks or blocks packaged as MQTT messages from a source like S3. You can have one or more files associated with a stream. The total size of a file associated with the stream cannot exceed more than 2 MB. The stream will be created with version 0. If a stream is created with the same streamID as a stream that existed and was deleted within last 90 days, we will resurrect that old stream by incrementing the version by 1.

Request syntax:

```
POST /streams/streamId
Content-type: application/json

{
  "description": "string",
  "files": [
    {
      "fileId": "integer",
      "s3Location": {
        "bucket": "string",
        "key": "string",
        "version": "string"
      }
    }
  ],
  "roleArn": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
streamId	StreamId	yes	The stream ID.

Request Body Parameters:

Name	Type	Req?	Description
description	StreamDescription	no	A description of the stream.
files	StreamFiles	yes	The files to stream.
roleArn	RoleArn	yes	An IAM role that allows the IoT service principal assumes to access your S3 files.

Response syntax:

```
Content-type: application/json

{
  "streamId": "string",
  "streamArn": "string",
  "description": "string",
  "streamVersion": "integer"
}
```

Response Body Parameters:

Name	Type	Req?	Description
streamId	StreamId	no	The stream ID.

Name	Type	Req?	Description
streamArn	StreamArn	no	The stream ARN.
description	StreamDescription	no	A description of the stream.
streamVersion	StreamVersion	no	The version of the stream.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ResourceAlreadyExistsException`

The resource already exists.

HTTP response code: 409

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  create-stream \
    --stream-id <value> \
    [--description <value>] \
```

```
    --files <value> \
    --role-arn <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "streamId": "string",
  "description": "string",
  "files": [
    {
      "fileId": "integer",
      "s3Location": {
        "bucket": "string",
        "key": "string",
        "version": "string"
      }
    }
  ],
  "roleArn": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
streamId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The stream ID.
description	string length max:2028 pattern: [^\\p{C}]+	A description of the stream.
files	list member: StreamFile	The files to stream.
StreamFile	StreamFile	
fileId	integer java class: java.lang.Integer range- max:255 min:0	The file ID.
s3Location	S3Location	The location of the file in S3.
bucket	string length min:1	The S3 bucket that contains the file to stream.
key	string length min:1	The name of the file within the S3 bucket to stream.
version	string	The file version.

Name	Type	Description
roleArn	string length max:2048 min:20	An IAM role that allows the IoT service principal assumes to access your S3 files.

Output:

```
{
  "streamId": "string",
  "streamArn": "string",
  "description": "string",
  "streamVersion": "integer"
}
```

cli output fields:

Name	Type	Description
streamId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The stream ID.
streamArn	string	The stream ARN.
description	string length max:2028 pattern: [^\\p{C}]+	A description of the stream.
streamVersion	integer java class: java.lang.Integer range- max:65535 min:0	The version of the stream.

CreateThing

Creates a thing record in the thing registry.

Request syntax:

```
POST /things/thingName
Content-type: application/json

{
  "thingTypeName": "string",
  "attributePayload": {
    "attributes": {
      "string": "string"
    },
    "merge": "boolean"
  }
}
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	yes	The name of the thing to create.

Request Body Parameters:

Name	Type	Req?	Description
thingTypeName	ThingTypeName	no	The name of the thing type associated with the new thing.
attributePayload	AttributePayload	no	The attribute payload, which consists of up to three name/value pairs in a JSON document. For example: *\"attributes\": {\"string1\": \"string2\"*}

Response syntax:

```
Content-type: application/json

{
  "thingName": "string",
  "thingArn": "string",
  "thingId": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
thingName	ThingName	no	The name of the new thing.
thingArn	ThingArn	no	The ARN of the new thing.
thingId	ThingId	no	The thing ID.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceAlreadyExistsException`

The resource already exists.

HTTP response code: 409

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  create-thing \
    --thing-name <value> \
    [--thing-type-name <value>] \
    [--attribute-payload <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string",
  "thingTypeName": "string",
  "attributePayload": {
    "attributes": {
      "string": "string"
    },
    "merge": "boolean"
  }
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string	The name of the thing to create.

Name	Type	Description
	length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
thingTypeName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing type associated with the new thing.
attributePayload	AttributePayload	The attribute payload, which consists of up to three name/value pairs in a JSON document. For example: `\"attributes\": {\"string1\": \"string2\"}`
attributes	map key: AttributeName value: AttributeValue	A JSON string containing up to three key-value pair in JSON format. For example: `\"attributes\": {\"string1\": \"string2\"}`
AttributeName	string length max:128 pattern: [a-zA-Z0-9_.,@/:#-]+	
AttributeValue	string length max:800 pattern: [a-zA-Z0-9_.,@/:#-]*	
merge	boolean	Specifies whether the list of attributes provided in the `AttributePayload` is merged with the attributes stored in the registry, instead of overwriting them. To remove an attribute, call `UpdateThing` with an empty attribute value. **Note** The `merge` attribute is only valid when calling `UpdateThing`.

Output:

```
{
```

```
  "thingName": "string",
  "thingArn": "string",
  "thingId": "string"
}
```

cli output fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the new thing.
thingArn	string	The ARN of the new thing.
thingId	string	The thing ID.

CreateThingGroup

Create a thing group.

Request syntax:

```
POST /thing-groups/thingGroupName
Content-type: application/json

{
  "parentGroupName": "string",
  "thingGroupProperties": {
    "thingGroupDescription": "string",
    "attributePayload": {
      "attributes": {
        "string": "string"
      },
      "merge": "boolean"
    }
  }
}
```

URI Request Parameters:

Name	Type	Req?	Description
thingGroupName	ThingGroupName	yes	The thing group name to create.

Request Body Parameters:

Name	Type	Req?	Description
parentGroupName	ThingGroupName	no	The name of the parent thing group.
thingGroupProperties	ThingGroupProperties	no	The thing group properties.

Response syntax:

```
Content-type: application/json

{
  "thingGroupName": "string",
  "thingGroupArn": "string",
  "thingGroupId": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
thingGroupName	ThingGroupName	no	The thing group name.
thingGroupArn	ThingGroupArn	no	The thing group ARN.
thingGroupId	ThingGroupId	no	The thing group ID.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceAlreadyExistsException`

The resource already exists.

HTTP response code: 409

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  create-thing-group \
    --thing-group-name <value> \
    [--parent-group-name <value>] \
    [--thing-group-properties <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingGroupName": "string",
  "parentGroupName": "string",
  "thingGroupProperties": {
    "thingGroupDescription": "string",
    "attributePayload": {
      "attributes": {
        "string": "string"
      },
      "merge": "boolean"
    }
  }
}
```

`cli-input-json` fields:

Name	Type	Description
thingGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The thing group name to create.
parentGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the parent thing group.
thingGroupProperties	ThingGroupProperties	The thing group properties.
thingGroupDescription	string length max:2028 pattern: [\\p{Graph}]*	The thing group description.
attributePayload	AttributePayload	The thing group attributes in JSON format.
attributes	map key: AttributeName value: AttributeValue	A JSON string containing up to three key-value pair in JSON format. For example: `\"attributes\": {\"string1\": \"string2\"}`
AttributeName	string length max:128 pattern: [a-zA-Z0-9_.,@/:#-]+	
AttributeValue	string length max:800 pattern: [a-zA-Z0-9_.,@/:#-]*	

Name	Type	Description
merge	boolean	Specifies whether the list of attributes provided in the `AttributePayload` is merged with the attributes stored in the registry, instead of overwriting them. To remove an attribute, call `UpdateThing` with an empty attribute value. **Note** The `merge` attribute is only valid when calling `UpdateThing`.

Output:

```
{
  "thingGroupName": "string",
  "thingGroupArn": "string",
  "thingGroupId": "string"
}
```

cli output fields:

Name	Type	Description
thingGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The thing group name.
thingGroupArn	string	The thing group ARN.
thingGroupId	string length max:128 min:1 pattern: [a-zA-Z0-9-]+	The thing group ID.

CreateThingType

Creates a new thing type.

Request syntax:

```
POST /thing-types/thingTypeName
Content-type: application/json

{
  "thingTypeProperties": {
    "thingTypeDescription": "string",
    "searchableAttributes": [
```

```
      "string"
    ]
  }
}
```

URI Request Parameters:

Name	Type	Req?	Description
thingTypeName	ThingTypeName	yes	The name of the thing type.

Request Body Parameters:

Name	Type	Req?	Description
thingTypeProperties	ThingTypeProperties	no	The ThingTypeProperties for the thing type to create. It contains information about the new thing type including a description, and a list of searchable thing attribute names.

Response syntax:

```
Content-type: application/json

{
  "thingTypeName": "string",
  "thingTypeArn": "string",
  "thingTypeId": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
thingTypeName	ThingTypeName	no	The name of the thing type.
thingTypeArn	ThingTypeArn	no	The Amazon Resource Name (ARN) of the thing type.
thingTypeId	ThingTypeId	no	The thing type ID.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceAlreadyExistsException`

The resource already exists.

HTTP response code: 409

CLI

Synopsis:

```
aws iot  create-thing-type \
    --thing-type-name <value> \
    [--thing-type-properties <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingTypeName": "string",
  "thingTypeProperties": {
    "thingTypeDescription": "string",
    "searchableAttributes": [
      "string"
    ]
  }
}
```

`cli-input-json` fields:

Name	Type	Description
thingTypeName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing type.

Name	Type	Description
thingTypeProperties	ThingTypeProperties	The ThingTypeProperties for the thing type to create. It contains information about the new thing type including a description, and a list of searchable thing attribute names.
thingTypeDescription	string length max:2028 pattern: [\\p{Graph}]*	The description of the thing type.
searchableAttributes	list member: AttributeName java class: java.util.List	A list of searchable thing attribute names.
AttributeName	string length max:128 pattern: [a-zA-Z0-9_.,@/:#-]+	

Output:

```
{
  "thingTypeName": "string",
  "thingTypeArn": "string",
  "thingTypeId": "string"
}
```

cli output fields:

Name	Type	Description
thingTypeName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing type.
thingTypeArn	string	The Amazon Resource Name (ARN) of the thing type.
thingTypeId	string	The thing type ID.

CreateTopicRule

Creates a rule. Creating rules is an administrator-level action. Any user who has permission to create rules will be able to access data processed by the rule.

Request syntax:

```
POST /rules/ruleName
Content-type: application/json

{
  "topicRulePayload": {
    "sql": "string",
    "description": "string",
    "actions": [
      {
        "dynamoDB": {
          "tableName": "string",
          "roleArn": "string",
          "operation": "string",
          "hashKeyField": "string",
          "hashKeyValue": "string",
          "hashKeyType": "string",
          "rangeKeyField": "string",
          "rangeKeyValue": "string",
          "rangeKeyType": "string",
          "payloadField": "string"
        },
        "dynamoDBv2": {
          "roleArn": "string",
          "putItem": {
            "tableName": "string"
          }
        },
        "lambda": {
          "functionArn": "string"
        },
        "sns": {
          "targetArn": "string",
          "roleArn": "string",
          "messageFormat": "string"
        },
        "sqs": {
          "roleArn": "string",
          "queueUrl": "string",
          "useBase64": "boolean"
        },
        "kinesis": {
          "roleArn": "string",
          "streamName": "string",
          "partitionKey": "string"
        },
        "republish": {
          "roleArn": "string",
          "topic": "string"
        },
        "s3": {
          "roleArn": "string",
          "bucketName": "string",
          "key": "string",
          "cannedAcl": "string"
        },
        "firehose": {
          "roleArn": "string",
          "deliveryStreamName": "string",
          "separator": "string"
        },
        "cloudwatchMetric": {
          "roleArn": "string",
          "metricNamespace": "string",
          "metricName": "string",
          "metricValue": "string",
```

```
        "metricUnit": "string",
        "metricTimestamp": "string"
      },
      "cloudwatchAlarm": {
        "roleArn": "string",
        "alarmName": "string",
        "stateReason": "string",
        "stateValue": "string"
      },
      "elasticsearch": {
        "roleArn": "string",
        "endpoint": "string",
        "index": "string",
        "type": "string",
        "id": "string"
      },
      "salesforce": {
        "token": "string",
        "url": "string"
      }
    }
  ],
  "ruleDisabled": "boolean",
  "awsIotSqlVersion": "string",
  "errorAction": {
    "dynamoDB": {
      "tableName": "string",
      "roleArn": "string",
      "operation": "string",
      "hashKeyField": "string",
      "hashKeyValue": "string",
      "hashKeyType": "string",
      "rangeKeyField": "string",
      "rangeKeyValue": "string",
      "rangeKeyType": "string",
      "payloadField": "string"
    },
    "dynamoDBv2": {
      "roleArn": "string",
      "putItem": {
        "tableName": "string"
      }
    },
    "lambda": {
      "functionArn": "string"
    },
    "sns": {
      "targetArn": "string",
      "roleArn": "string",
      "messageFormat": "string"
    },
    "sqs": {
      "roleArn": "string",
      "queueUrl": "string",
      "useBase64": "boolean"
    },
    "kinesis": {
      "roleArn": "string",
      "streamName": "string",
      "partitionKey": "string"
    },
    "republish": {
      "roleArn": "string",
      "topic": "string"
    },
    "s3": {
```

```
        "roleArn": "string",
        "bucketName": "string",
        "key": "string",
        "cannedAcl": "string"
      },
      "firehose": {
        "roleArn": "string",
        "deliveryStreamName": "string",
        "separator": "string"
      },
      "cloudwatchMetric": {
        "roleArn": "string",
        "metricNamespace": "string",
        "metricName": "string",
        "metricValue": "string",
        "metricUnit": "string",
        "metricTimestamp": "string"
      },
      "cloudwatchAlarm": {
        "roleArn": "string",
        "alarmName": "string",
        "stateReason": "string",
        "stateValue": "string"
      },
      "elasticsearch": {
        "roleArn": "string",
        "endpoint": "string",
        "index": "string",
        "type": "string",
        "id": "string"
      },
      "salesforce": {
        "token": "string",
        "url": "string"
      }
    }
  }
}
```

URI Request Parameters:

Name	Type	Req?	Description
ruleName	RuleName	yes	The name of the rule.

Request Body Parameters:

Name	Type	Req?	Description
topicRulePayload	TopicRulePayload	yes	The rule payload.

Errors:

`SqlParseException`

The Rule-SQL expression can't be parsed correctly.

HTTP response code: 400

`InternalException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceAlreadyExistsException`

The resource already exists.

HTTP response code: 409

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  create-topic-rule \
    --rule-name <value> \
    --topic-rule-payload <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "ruleName": "string",
  "topicRulePayload": {
    "sql": "string",
    "description": "string",
    "actions": [
      {
        "dynamoDB": {
          "tableName": "string",
          "roleArn": "string",
          "operation": "string",
          "hashKeyField": "string",
          "hashKeyValue": "string",
          "hashKeyType": "string",
          "rangeKeyField": "string",
          "rangeKeyValue": "string",
          "rangeKeyType": "string",
          "payloadField": "string"
        },
        "dynamoDBv2": {
          "roleArn": "string",
          "putItem": {
            "tableName": "string"
          }
        },
        "lambda": {
          "functionArn": "string"
        },
        "sns": {
          "targetArn": "string",
```

```
        "roleArn": "string",
        "messageFormat": "string"
      },
      "sqs": {
        "roleArn": "string",
        "queueUrl": "string",
        "useBase64": "boolean"
      },
      "kinesis": {
        "roleArn": "string",
        "streamName": "string",
        "partitionKey": "string"
      },
      "republish": {
        "roleArn": "string",
        "topic": "string"
      },
      "s3": {
        "roleArn": "string",
        "bucketName": "string",
        "key": "string",
        "cannedAcl": "string"
      },
      "firehose": {
        "roleArn": "string",
        "deliveryStreamName": "string",
        "separator": "string"
      },
      "cloudwatchMetric": {
        "roleArn": "string",
        "metricNamespace": "string",
        "metricName": "string",
        "metricValue": "string",
        "metricUnit": "string",
        "metricTimestamp": "string"
      },
      "cloudwatchAlarm": {
        "roleArn": "string",
        "alarmName": "string",
        "stateReason": "string",
        "stateValue": "string"
      },
      "elasticsearch": {
        "roleArn": "string",
        "endpoint": "string",
        "index": "string",
        "type": "string",
        "id": "string"
      },
      "salesforce": {
        "token": "string",
        "url": "string"
      }
    }
  ],
  "ruleDisabled": "boolean",
  "awsIotSqlVersion": "string",
  "errorAction": {
    "dynamoDB": {
      "tableName": "string",
      "roleArn": "string",
      "operation": "string",
      "hashKeyField": "string",
      "hashKeyValue": "string",
      "hashKeyType": "string",
      "rangeKeyField": "string",
```

```
    "rangeKeyValue": "string",
    "rangeKeyType": "string",
    "payloadField": "string"
  },
  "dynamoDBv2": {
    "roleArn": "string",
    "putItem": {
      "tableName": "string"
    }
  },
  "lambda": {
    "functionArn": "string"
  },
  "sns": {
    "targetArn": "string",
    "roleArn": "string",
    "messageFormat": "string"
  },
  "sqs": {
    "roleArn": "string",
    "queueUrl": "string",
    "useBase64": "boolean"
  },
  "kinesis": {
    "roleArn": "string",
    "streamName": "string",
    "partitionKey": "string"
  },
  "republish": {
    "roleArn": "string",
    "topic": "string"
  },
  "s3": {
    "roleArn": "string",
    "bucketName": "string",
    "key": "string",
    "cannedAcl": "string"
  },
  "firehose": {
    "roleArn": "string",
    "deliveryStreamName": "string",
    "separator": "string"
  },
  "cloudwatchMetric": {
    "roleArn": "string",
    "metricNamespace": "string",
    "metricName": "string",
    "metricValue": "string",
    "metricUnit": "string",
    "metricTimestamp": "string"
  },
  "cloudwatchAlarm": {
    "roleArn": "string",
    "alarmName": "string",
    "stateReason": "string",
    "stateValue": "string"
  },
  "elasticsearch": {
    "roleArn": "string",
    "endpoint": "string",
    "index": "string",
    "type": "string",
    "id": "string"
  },
  "salesforce": {
    "token": "string",
```

```
        "url": "string"
      }
    }
  }
}
```

`cli-input-json` fields:

Name	Type	Description
ruleName	string length max:128 min:1 pattern: ^[a-zA-Z0-9_]+$	The name of the rule.
topicRulePayload	TopicRulePayload	The rule payload.
sql	string	The SQL statement used to query the topic. For more information, see AWS IoT SQL Reference in the *AWS IoT Developer Guide*.
description	string	The description of the rule.
actions	list member: Action	The actions associated with the rule.
Action	Action	
dynamoDB	DynamoDBAction	Write to a DynamoDB table.
tableName	string	The name of the DynamoDB table.
roleArn	string	The ARN of the IAM role that grants access to the DynamoDB table.
operation	string	The type of operation to be performed. This follows the substitution template, so it can be `$ operation`, but the substitution must result in one of the following: `INSERT`, `UPDATE`, or `DELETE`.
hashKeyField	string	The hash key name.
hashKeyValue	string	The hash key value.
hashKeyType	string enum: STRING | NUMBER java class: iot.goldeneye.service.DynamoKeyType	The hash key type. Valid values are "STRING" or "NUMBER"
rangeKeyField	string	The range key name.

Name	Type	Description
rangeKeyValue	string	The range key value.
rangeKeyType	string enum: STRING | NUMBER java class: iot.goldeneye.service.DynamoKeyType	The range key type. Valid values are "STRING" or "NUMBER"
payloadField	string	The action payload. This name can be customized.
dynamoDBv2	DynamoDBv2Action	Write to a DynamoDB table. This is a new version of the DynamoDB action. It allows you to write each attribute in an MQTT message payload into a separate DynamoDB column.
roleArn	string	The ARN of the IAM role that grants access to the DynamoDB table.
putItem	PutItemInput	Specifies the DynamoDB table to which the message data will be written. For example: `{ "dynamoDBv2": { "roleArn": "aws:iam:12341251:my-role" "putItem": { "tableName": "my-table" } } }` Each attribute in the message payload will be written to a separate column in the DynamoDB database.
tableName	string	The table where the message data will be written
lambda	LambdaAction	Invoke a Lambda function.
functionArn	string	The ARN of the Lambda function.
sns	SnsAction	Publish to an Amazon SNS topic.
targetArn	string	The ARN of the SNS topic.
roleArn	string	The ARN of the IAM role that grants access.

Name	Type	Description
messageFormat	string enum: RAW | JSON java class: iot.goldeneye.service.MessageFormat	The message format of the message to publish. Optional. Accepted values are "JSON" and "RAW". The default value of the attribute is "RAW". SNS uses this setting to determine if the payload should be parsed and relevant platform-specific bits of the payload should be extracted. To read more about SNS message formats, see http://docs.aws.amazon.com/sns/latest/dg/json-formats.html refer to their official documentation.
sqs	SqsAction	Publish to an Amazon SQS queue.
roleArn	string	The ARN of the IAM role that grants access.
queueUrl	string	The URL of the Amazon SQS queue.
useBase64	boolean java class: java.lang.Boolean	Specifies whether to use Base64 encoding.
kinesis	KinesisAction	Write data to an Amazon Kinesis stream.
roleArn	string	The ARN of the IAM role that grants access to the Amazon Kinesis stream.
streamName	string	The name of the Amazon Kinesis stream.
partitionKey	string	The partition key.
republish	RepublishAction	Publish to another MQTT topic.
roleArn	string	The ARN of the IAM role that grants access.
topic	string	The name of the MQTT topic.
s3	S3Action	Write to an Amazon S3 bucket.
roleArn	string	The ARN of the IAM role that grants access.
bucketName	string	The Amazon S3 bucket.
key	string	The object key.

Name	Type	Description
cannedAcl	string enum: private | public-read | public-read-write | aws-exec-read | authenticated-read | bucket-owner-read | bucket-owner-full-control | log-delivery-write java class: iot.goldeneye.service.CannedAccessControlList	The Amazon S3 canned ACL that controls access to the object identified by the object key. For more information, see S3 canned ACLs.
firehose	FirehoseAction	Write to an Amazon Kinesis Firehose stream.
roleArn	string	The IAM role that grants access to the Amazon Kinesis Firehose stream.
deliveryStreamName	string	The delivery stream name.
separator	string pattern: ([ ])|( )|(,)	A character separator that will be used to separate records written to the Firehose stream. Valid values are: '\n' (newline), '\t' (tab), '\r\n' (Windows newline), ',' (comma).
cloudwatchMetric	CloudwatchMetricAction	Capture a CloudWatch metric.
roleArn	string	The IAM role that allows access to the CloudWatch metric.
metricNamespace	string	The CloudWatch metric namespace name.
metricName	string	The CloudWatch metric name.
metricValue	string	The CloudWatch metric value.
metricUnit	string	The metric unit supported by CloudWatch.
metricTimestamp	string	An optional Unix timestamp.
cloudwatchAlarm	CloudwatchAlarmAction	Change the state of a CloudWatch alarm.
roleArn	string	The IAM role that allows access to the CloudWatch alarm.
alarmName	string	The CloudWatch alarm name.
stateReason	string	The reason for the alarm change.

Name	Type	Description
stateValue	string	The value of the alarm state. Acceptable values are: OK, ALARM, INSUFFICIENT_DATA.
elasticsearch	ElasticsearchAction	Write data to an Amazon Elasticsearch Service domain.
roleArn	string	The IAM role ARN that has access to Elasticsearch.
endpoint	string pattern: https?://.*	The endpoint of your Elasticsearch domain.
index	string	The Elasticsearch index where you want to store your data.
type	string	The type of document you are storing.
id	string	The unique identifier for the document you are storing.
salesforce	SalesforceAction	Send a message to a Salesforce IoT Cloud Input Stream.
token	string length min:40	The token used to authenticate access to the Salesforce IoT Cloud Input Stream. The token is available from the Salesforce IoT Cloud platform after creation of the Input Stream.
url	string length max:2000 pattern: https://ingestion-[a-zA-Z0-9]{1,12}.[a-zA-Z0-9]+.((sfdc-matrix.net)|(sfdcnow.com))/streams/w *1,20*/w *1,20*/event	The URL exposed by the Salesforce IoT Cloud Input Stream. The URL is available from the Salesforce IoT Cloud platform after creation of the Input Stream.
ruleDisabled	boolean java class: java.lang.Boolean	Specifies whether the rule is disabled.
awsIotSqlVersion	string	The version of the SQL rules engine to use when evaluating the rule.
errorAction	Action	The action to take when an error occurs.
dynamoDB	DynamoDBAction	Write to a DynamoDB table.
tableName	string	The name of the DynamoDB table.

Name	Type	Description
roleArn	string	The ARN of the IAM role that grants access to the DynamoDB table.
operation	string	The type of operation to be performed. This follows the substitution template, so it can be `$ operation`, but the substitution must result in one of the following: `INSERT`, `UPDATE`, or `DELETE`.
hashKeyField	string	The hash key name.
hashKeyValue	string	The hash key value.
hashKeyType	string enum: STRING | NUMBER java class: iot.goldeneye.service.DynamoKeyType	The hash key type. Valid values are "STRING" or "NUMBER"
rangeKeyField	string	The range key name.
rangeKeyValue	string	The range key value.
rangeKeyType	string enum: STRING | NUMBER java class: iot.goldeneye.service.DynamoKeyType	The range key type. Valid values are "STRING" or "NUMBER"
payloadField	string	The action payload. This name can be customized.
dynamoDBv2	DynamoDBv2Action	Write to a DynamoDB table. This is a new version of the DynamoDB action. It allows you to write each attribute in an MQTT message payload into a separate DynamoDB column.
roleArn	string	The ARN of the IAM role that grants access to the DynamoDB table.

Name	Type	Description
putItem	PutItemInput	Specifies the DynamoDB table to which the message data will be written. For example: `{ "dynamoDBv2": { "roleArn": "aws:iam:12341251:my-role" "putItem": { "tableName": "my-table" } } }` Each attribute in the message payload will be written to a separate column in the DynamoDB database.
tableName	string	The table where the message data will be written
lambda	LambdaAction	Invoke a Lambda function.
functionArn	string	The ARN of the Lambda function.
sns	SnsAction	Publish to an Amazon SNS topic.
targetArn	string	The ARN of the SNS topic.
roleArn	string	The ARN of the IAM role that grants access.
messageFormat	string enum: RAW | JSON java class: iot.goldeneye.service.MessageFormat	The message format of the message to publish. Optional. Accepted values are "JSON" and "RAW". The default value of the attribute is "RAW". SNS uses this setting to determine if the payload should be parsed and relevant platform-specific bits of the payload should be extracted. To read more about SNS message formats, see http://docs.aws.amazon.com/sns/latest/dg/json-formats.html refer to their official documentation.
sqs	SqsAction	Publish to an Amazon SQS queue.
roleArn	string	The ARN of the IAM role that grants access.
queueUrl	string	The URL of the Amazon SQS queue.

Name	Type	Description
useBase64	boolean java class: java.lang.Boolean	Specifies whether to use Base64 encoding.
kinesis	KinesisAction	Write data to an Amazon Kinesis stream.
roleArn	string	The ARN of the IAM role that grants access to the Amazon Kinesis stream.
streamName	string	The name of the Amazon Kinesis stream.
partitionKey	string	The partition key.
republish	RepublishAction	Publish to another MQTT topic.
roleArn	string	The ARN of the IAM role that grants access.
topic	string	The name of the MQTT topic.
s3	S3Action	Write to an Amazon S3 bucket.
roleArn	string	The ARN of the IAM role that grants access.
bucketName	string	The Amazon S3 bucket.
key	string	The object key.
cannedAcl	string enum: private | public-read | public-read-write | aws-exec-read | authenticated-read | bucket-owner-read | bucket-owner-full-control | log-delivery-write java class: iot.goldeneye.service.CannedAccessControlList	The Amazon S3 canned ACL that controls access to the object identified by the object key. For more information, see S3 canned ACLs.
firehose	FirehoseAction	Write to an Amazon Kinesis Firehose stream.
roleArn	string	The IAM role that grants access to the Amazon Kinesis Firehose stream.
deliveryStreamName	string	The delivery stream name.
separator	string pattern: ([ ])|( )|(,)	A character separator that will be used to separate records written to the Firehose stream. Valid values are: '\n' (newline), '\t' (tab), '\r\n' (Windows newline), ',' (comma).

Name	Type	Description
cloudwatchMetric	CloudwatchMetricAction	Capture a CloudWatch metric.
roleArn	string	The IAM role that allows access to the CloudWatch metric.
metricNamespace	string	The CloudWatch metric namespace name.
metricName	string	The CloudWatch metric name.
metricValue	string	The CloudWatch metric value.
metricUnit	string	The metric unit supported by CloudWatch.
metricTimestamp	string	An optional Unix timestamp.
cloudwatchAlarm	CloudwatchAlarmAction	Change the state of a CloudWatch alarm.
roleArn	string	The IAM role that allows access to the CloudWatch alarm.
alarmName	string	The CloudWatch alarm name.
stateReason	string	The reason for the alarm change.
stateValue	string	The value of the alarm state. Acceptable values are: OK, ALARM, INSUFFICIENT_DATA.
elasticsearch	ElasticsearchAction	Write data to an Amazon Elasticsearch Service domain.
roleArn	string	The IAM role ARN that has access to Elasticsearch.
endpoint	string pattern: https?://.*	The endpoint of your Elasticsearch domain.
index	string	The Elasticsearch index where you want to store your data.
type	string	The type of document you are storing.
id	string	The unique identifier for the document you are storing.
salesforce	SalesforceAction	Send a message to a Salesforce IoT Cloud Input Stream.

Name	Type	Description
token	string length min:40	The token used to authenticate access to the Salesforce IoT Cloud Input Stream. The token is available from the Salesforce IoT Cloud platform after creation of the Input Stream.
url	string length max:2000 pattern: https://ingestion-[a-zA-Z0-9]{1,12}.[a-zA-Z0-9]+.((sfdc-matrix.net)|(sfdcnow.com))/streams/w *1,20*/w *1,20*/event	The URL exposed by the Salesforce IoT Cloud Input Stream. The URL is available from the Salesforce IoT Cloud platform after creation of the Input Stream.

Output:

None

DeleteAuthorizer

Deletes an authorizer.

Request syntax:

```
DELETE /authorizer/authorizerName
```

URI Request Parameters:

Name	Type	Req?	Description
authorizerName	AuthorizerName	yes	The name of the authorizer to delete.

Errors:

`DeleteConflictException`

You can't delete the resource because it is attached to one or more resources.

HTTP response code: 409

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  delete-authorizer \
    --authorizer-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "authorizerName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
authorizerName	string length max:128 min:1 pattern: [w=,@-]+	The name of the authorizer to delete.

Output:

None

DeleteCACertificate

Deletes a registered CA certificate.

Request syntax:

```
DELETE /cacertificate/caCertificateId
```

URI Request Parameters:

Name	Type	Req?	Description
certificateId	CertificateId	yes	The ID of the certificate to delete. (The last part of the certificate ARN contains the certificate ID.)

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`CertificateStateException`

The certificate operation is not allowed.

HTTP response code: 406

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  delete-ca-certificate \
    --certificate-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "certificateId": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the certificate to delete. (The last part of the certificate ARN contains the certificate ID.)

Output:

None

DeleteCertificate

Deletes the specified certificate.

A certificate cannot be deleted if it has a policy attached to it or if its status is set to ACTIVE. To delete a certificate, first use the DetachPrincipalPolicy API to detach all policies. Next, use the UpdateCertificate API to set the certificate to the INACTIVE status.

Request syntax:

```
DELETE /certificates/certificateId?forceDelete=forceDelete
```

URI Request Parameters:

Name	Type	Req?	Description
certificateId	CertificateId	yes	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)
forceDelete	ForceDelete	no	Forces a certificate request to be deleted.

Errors:

`CertificateStateException`

The certificate operation is not allowed.

HTTP response code: 406

`DeleteConflictException`

You can't delete the resource because it is attached to one or more resources.

HTTP response code: 409

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  delete-certificate \
    --certificate-id <value> \
    [--force-delete | --no-force-delete]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "certificateId": "string",
  "forceDelete": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)
forceDelete	boolean	Forces a certificate request to be deleted.

Output:

None

DeleteOTAUpdate

Delete an OTA update.

Request syntax:

```
DELETE /otaUpdates/otaUpdateId
```

URI Request Parameters:

Name	Type	Req?	Description
otaUpdateId	OTAUpdateId	yes	The OTA update ID to delete.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  delete-ota-update \
    --ota-update-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "otaUpdateId": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
otaUpdateId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The OTA update ID to delete.

Output:

None

DeletePolicy

Deletes the specified policy.

A policy cannot be deleted if it has non-default versions or it is attached to any certificate.

To delete a policy, use the DeletePolicyVersion API to delete all non-default versions of the policy; use the DetachPrincipalPolicy API to detach the policy from any certificate; and then use the DeletePolicy API to delete the policy.

When a policy is deleted using DeletePolicy, its default version is deleted with it.

Request syntax:

```
DELETE /policies/policyName
```

URI Request Parameters:

Name	Type	Req?	Description
policyName	PolicyName	yes	The name of the policy to delete.

Errors:

`DeleteConflictException`

You can't delete the resource because it is attached to one or more resources.

HTTP response code: 409

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  delete-policy \
    --policy-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "policyName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The name of the policy to delete.

Output:

None

DeletePolicyVersion

Deletes the specified version of the specified policy. You cannot delete the default version of a policy using this API. To delete the default version of a policy, use DeletePolicy. To find out which version of a policy is marked as the default version, use ListPolicyVersions.

Request syntax:

```
DELETE /policies/policyName/version/policyVersionId
```

URI Request Parameters:

Name	Type	Req?	Description
policyName	PolicyName	yes	The name of the policy.
policyVersionId	PolicyVersionId	yes	The policy version ID.

Errors:

`DeleteConflictException`

You can't delete the resource because it is attached to one or more resources.

HTTP response code: 409

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  delete-policy-version \
    --policy-name <value> \
    --policy-version-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "policyName": "string",
  "policyVersionId": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The name of the policy.

Name	Type	Description
policyVersionId	string pattern: [0-9]+	The policy version ID.

Output:

None

DeleteRegistrationCode

Deletes a CA certificate registration code.

Request syntax:

```
DELETE /registrationcode
```

Errors:

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  delete-registration-code  \
    [--cli-input-json <value>] \
```

```
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
}
```

Output:

None

DeleteRoleAlias

Deletes a role alias

Request syntax:

```
DELETE /role-aliases/roleAlias
```

URI Request Parameters:

Name	Type	Req?	Description
roleAlias	RoleAlias	yes	The role alias to delete.

Errors:

`DeleteConflictException`

You can't delete the resource because it is attached to one or more resources.

HTTP response code: 409

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  delete-role-alias \
    --role-alias <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "roleAlias": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
roleAlias	string length max:128 min:1 pattern: [w=,@-]+	The role alias to delete.

Output:

None

DeleteStream

Deletes a stream.

Request syntax:

```
DELETE /streams/streamId
```

URI Request Parameters:

Name	Type	Req?	Description
streamId	StreamId	yes	The stream ID.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`DeleteConflictException`

You can't delete the resource because it is attached to one or more resources.

HTTP response code: 409

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  delete-stream \
    --stream-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "streamId": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
streamId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The stream ID.

Output:

None

DeleteThing

Deletes the specified thing.

Request syntax:

```
DELETE /things/thingName?expectedVersion=expectedVersion
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	yes	The name of the thing to delete.
expectedVersion	OptionalVersion	no	The expected version of the thing record in the registry. If the version of the record in the registry does not match the expected version specified in the request, the `DeleteThing` request is rejected with a `VersionConflictException`.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`VersionConflictException`

An exception thrown when the version of a thing passed to a command is different than the version specified with the --version parameter.

HTTP response code: 409

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  delete-thing \
    --thing-name <value> \
    [--expected-version <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string",
  "expectedVersion": "long"
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing to delete.

Name	Type	Description
expectedVersion	long java class: java.lang.Long	The expected version of the thing record in the registry. If the version of the record in the registry does not match the expected version specified in the request, the `DeleteThing` request is rejected with a `VersionConflictException`.

Output:

None

DeleteThingGroup

Deletes a thing group.

Request syntax:

```
DELETE /thing-groups/thingGroupName?expectedVersion=expectedVersion
```

URI Request Parameters:

Name	Type	Req?	Description
thingGroupName	ThingGroupName	yes	The name of the thing group to delete.
expectedVersion	OptionalVersion	no	The expected version of the thing group to delete.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`VersionConflictException`

An exception thrown when the version of a thing passed to a command is different than the version specified with the --version parameter.

HTTP response code: 409

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  delete-thing-group \
    --thing-group-name <value> \
    [--expected-version <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingGroupName": "string",
  "expectedVersion": "long"
}
```

`cli-input-json` fields:

Name	Type	Description
thingGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing group to delete.
expectedVersion	long java class: java.lang.Long	The expected version of the thing group to delete.

Output:

None

DeleteThingShadow

Deletes the thing shadow for the specified thing.

For more information, see DeleteThingShadow in the AWS IoT Developer Guide.

Request syntax:

```
DELETE /things/thingName/shadow
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	yes	The name of the thing.

Response syntax:

```
Content-type: application/json

{
  "payload": "blob"
}
```

Response Body Parameters:

Name	Type	Req?	Description
payload	JsonDocument	yes	The state information, in JSON format.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`MethodNotAllowedException`

The specified combination of HTTP verb and URI is not supported.

HTTP response code: 405

`UnsupportedDocumentEncodingException`

The encoding is not supported.

HTTP response code: 415

CLI

Synopsis:

```
aws iot  delete-thing-shadow \
    --thing-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing.

Output:

```
{
  "payload": "blob"
}
```

cli output fields:

Name	Type	Description
payload	blob	The state information, in JSON format.

DeleteThingType

Deletes the specified thing type . You cannot delete a thing type if it has things associated with it. To delete a thing type, first mark it as deprecated by calling DeprecateThingType, then remove any associated things by calling UpdateThing to change the thing type on any associated thing, and finally use DeleteThingType to delete the thing type.

Request syntax:

```
DELETE /thing-types/thingTypeName
```

URI Request Parameters:

Name	Type	Req?	Description
thingTypeName	ThingTypeName	yes	The name of the thing type.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  delete-thing-type \
    --thing-type-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingTypeName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
thingTypeName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing type.

Output:

None

DeleteTopicRule

Deletes the rule.

Request syntax:

```
DELETE /rules/ruleName
```

URI Request Parameters:

Name	Type	Req?	Description
ruleName	RuleName	no	The name of the rule.

Errors:

`InternalException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

CLI

Synopsis:

```
aws iot  delete-topic-rule \
    [--rule-name <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "ruleName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
ruleName	string length max:128 min:1 pattern: ^[a-zA-Z0-9_]+$	The name of the rule.

Output:

None

DeleteV2LoggingLevel

Deletes a logging level.

Request syntax:

```
DELETE /v2LoggingLevel?targetName=targetName&targetType=targetType
```

URI Request Parameters:

Name	Type	Req?	Description
targetType	LogTargetType	yes	The type of resource for which you are configuring logging. Must be `THING_Group`.
targetName	LogTargetName	yes	The name of the resource for which you are configuring logging.

Errors:

`InternalException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  delete-v2-logging-level \
    --target-type <value> \
    --target-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "targetType": "string",
  "targetName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
targetType	string enum: DEFAULT | THING_GROUP java class: iot.goldeneye.service.LogTargetType	The type of resource for which you are configuring logging. Must be `THING_Group`.
targetName	string	The name of the resource for which you are configuring logging.

Output:

None

DeprecateThingType

Deprecates a thing type. You can not associate new things with deprecated thing type.

Request syntax:

```
POST /thing-types/thingTypeName/deprecate
Content-type: application/json

{
  "undoDeprecate": "boolean"
}
```

URI Request Parameters:

Name	Type	Req?	Description
thingTypeName	ThingTypeName	yes	The name of the thing type to deprecate.

Request Body Parameters:

Name	Type	Req?	Description
undoDeprecate	UndoDeprecate	no	Whether to undeprecate a deprecated thing type. If **true**, the thing type will not be deprecated anymore and you can associate it with things.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  deprecate-thing-type \
    --thing-type-name <value> \
    [--undo-deprecate | --no-undo-deprecate]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingTypeName": "string",
  "undoDeprecate": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
thingTypeName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing type to deprecate.
undoDeprecate	boolean	Whether to undeprecate a deprecated thing type. If **true**, the thing type will not be deprecated anymore and you can associate it with things.

Output:

None

DescribeAuthorizer

Describes an authorizer.

Request syntax:

```
GET /authorizer/authorizerName
```

URI Request Parameters:

Name	Type	Req?	Description
authorizerName	AuthorizerName	yes	The name of the authorizer to describe.

Response syntax:

```
Content-type: application/json

{
  "authorizerDescription": {
    "authorizerName": "string",
    "authorizerArn": "string",
    "authorizerFunctionArn": "string",
    "tokenKeyName": "string",
    "tokenSigningPublicKeys": {
      "string": "string"
    },
    "status": "string",
    "creationDate": "timestamp",
    "lastModifiedDate": "timestamp"
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
authorizerDescription	AuthorizerDescription	no	The authorizer description.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  describe-authorizer \
    --authorizer-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "authorizerName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
authorizerName	string length max:128 min:1 pattern: [w=,@-]+	The name of the authorizer to describe.

Output:

```
{
  "authorizerDescription": {
    "authorizerName": "string",
    "authorizerArn": "string",
    "authorizerFunctionArn": "string",
    "tokenKeyName": "string",
    "tokenSigningPublicKeys": {
      "string": "string"
    },
    "status": "string",
    "creationDate": "timestamp",
    "lastModifiedDate": "timestamp"
  }
```

```
}
```

cli output fields:

Name	Type	Description
authorizerDescription	AuthorizerDescription	The authorizer description.
authorizerName	string length max:128 min:1 pattern: [w=,@-]+	The authorizer name.
authorizerArn	string	The authorizer ARN.
authorizerFunctionArn	string	The authorizer's Lambda function ARN.
tokenKeyName	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The key used to extract the token from the HTTP headers.
tokenSigningPublicKeys	map key: KeyName value: KeyValue	The public keys used to validate the token signature returned by your custom authentication service.
KeyName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
KeyValue	string length max:5120	
status	string enum: ACTIVE | INACTIVE java class: iot.identity.service.AuthorizerStatus	The status of the authorizer.
creationDate	timestamp	The UNIX timestamp of when the authorizer was created.
lastModifiedDate	timestamp	The UNIX timestamp of when the authorizer was last updated.

DescribeCACertificate

Describes a registered CA certificate.

Request syntax:

```
GET /cacertificate/caCertificateId
```

URI Request Parameters:

Name	Type	Req?	Description
certificateId	CertificateId	yes	The CA certificate identifier.

Response syntax:

```
Content-type: application/json

{
  "certificateDescription": {
    "certificateArn": "string",
    "certificateId": "string",
    "status": "string",
    "certificatePem": "string",
    "ownedBy": "string",
    "creationDate": "timestamp",
    "autoRegistrationStatus": "string"
  },
  "registrationConfig": {
    "templateBody": "string",
    "roleArn": "string"
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
certificateDescription	CACertificateDescription	no	The CA certificate description.
registrationConfig	RegistrationConfig	no	Information about the registration configuration.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  describe-ca-certificate \
    --certificate-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "certificateId": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The CA certificate identifier.

Output:

```
{
  "certificateDescription": {
    "certificateArn": "string",
    "certificateId": "string",
    "status": "string",
    "certificatePem": "string",
    "ownedBy": "string",
    "creationDate": "timestamp",
    "autoRegistrationStatus": "string"
  },
```

```
  "registrationConfig": {
    "templateBody": "string",
    "roleArn": "string"
  }
}
```

cli output fields:

Name	Type	Description
certificateDescription	CACertificateDescription	The CA certificate description.
certificateArn	string	The CA certificate ARN.
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The CA certificate ID.
status	string enum: ACTIVE | INACTIVE java class: iot.identity.service.CACertificateStatus	The status of a CA certificate.
certificatePem	string length max:65536 min:1	The CA certificate data, in PEM format.
ownedBy	string pattern: [0-9]{12}	The owner of the CA certificate.
creationDate	timestamp	The date the CA certificate was created.
autoRegistrationStatus	string enum: ENABLE | DISABLE java class: iot.identity.service.AutoRegistrationStatus	Whether the CA certificate configured for auto registration of device certificates. Valid values are "ENABLE" and "DISABLE"
registrationConfig	RegistrationConfig	Information about the registration configuration.
templateBody	string	The template body.
roleArn	string length max:2048 min:20	The ARN of the role.

DescribeCertificate

Gets information about the specified certificate.

Request syntax:

```
GET /certificates/certificateId
```

URI Request Parameters:

Name	Type	Req?	Description
certificateId	CertificateId	yes	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)

Response syntax:

```
Content-type: application/json

{
  "certificateDescription": {
    "certificateArn": "string",
    "certificateId": "string",
    "caCertificateId": "string",
    "status": "string",
    "certificatePem": "string",
    "ownedBy": "string",
    "previousOwnedBy": "string",
    "creationDate": "timestamp",
    "lastModifiedDate": "timestamp",
    "transferData": {
      "transferMessage": "string",
      "rejectReason": "string",
      "transferDate": "timestamp",
      "acceptDate": "timestamp",
      "rejectDate": "timestamp"
    }
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
certificateDescription	CertificateDescription	no	The description of the certificate.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  describe-certificate \
    --certificate-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "certificateId": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)

Output:

```
{
  "certificateDescription": {
    "certificateArn": "string",
    "certificateId": "string",
    "caCertificateId": "string",
    "status": "string",
    "certificatePem": "string",
```

```
    "ownedBy": "string",
    "previousOwnedBy": "string",
    "creationDate": "timestamp",
    "lastModifiedDate": "timestamp",
    "transferData": {
      "transferMessage": "string",
      "rejectReason": "string",
      "transferDate": "timestamp",
      "acceptDate": "timestamp",
      "rejectDate": "timestamp"
    }
  }
}
```

cli output fields:

Name	Type	Description
certificateDescription	CertificateDescription	The description of the certificate.
certificateArn	string	The ARN of the certificate.
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the certificate.
caCertificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The certificate ID of the CA certificate used to sign this certificate.
status	string enum: ACTIVE | INACTIVE | REVOKED | PENDING_TRANSFER | REGISTER_INACTIVE | PENDING_ACTIVATION java class: iot.identity.service.CertificateStatus	The status of the certificate.
certificatePem	string length max:65536 min:1	The certificate data, in PEM format.
ownedBy	string pattern: [0-9]{12}	The ID of the AWS account that owns the certificate.
previousOwnedBy	string pattern: [0-9]{12}	The ID of the AWS account of the previous owner of the certificate.
creationDate	timestamp	The date and time the certificate was created.
lastModifiedDate	timestamp	The date and time the certificate was last modified.

Name	Type	Description
transferData	TransferData	The transfer data.
transferMessage	string length max:128	The transfer message.
rejectReason	string length max:128	The reason why the transfer was rejected.
transferDate	timestamp	The date the transfer took place.
acceptDate	timestamp	The date the transfer was accepted.
rejectDate	timestamp	The date the transfer was rejected.

DescribeDefaultAuthorizer

Describes the default authorizer.

Request syntax:

```
GET /default-authorizer
```

Response syntax:

```
Content-type: application/json

{
  "authorizerDescription": {
    "authorizerName": "string",
    "authorizerArn": "string",
    "authorizerFunctionArn": "string",
    "tokenKeyName": "string",
    "tokenSigningPublicKeys": {
      "string": "string"
    },
    "status": "string",
    "creationDate": "timestamp",
    "lastModifiedDate": "timestamp"
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
authorizerDescription	AuthorizerDescription	no	The default authorizer's description.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  describe-default-authorizer  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
}
```

Output:

```
{
  "authorizerDescription": {
    "authorizerName": "string",
    "authorizerArn": "string",
    "authorizerFunctionArn": "string",
```

```
    "tokenKeyName": "string",
    "tokenSigningPublicKeys": {
      "string": "string"
    },
    "status": "string",
    "creationDate": "timestamp",
    "lastModifiedDate": "timestamp"
  }
}
```

cli output fields:

Name	Type	Description
authorizerDescription	AuthorizerDescription	The default authorizer's description.
authorizerName	string length max:128 min:1 pattern: [w=,@-]+	The authorizer name.
authorizerArn	string	The authorizer ARN.
authorizerFunctionArn	string	The authorizer's Lambda function ARN.
tokenKeyName	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The key used to extract the token from the HTTP headers.
tokenSigningPublicKeys	map key: KeyName value: KeyValue	The public keys used to validate the token signature returned by your custom authentication service.
KeyName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
KeyValue	string length max:5120	
status	string enum: ACTIVE | INACTIVE java class: iot.identity.service.AuthorizerStatus	The status of the authorizer.
creationDate	timestamp	The UNIX timestamp of when the authorizer was created.
lastModifiedDate	timestamp	The UNIX timestamp of when the authorizer was last updated.

DescribeEndpoint

Returns a unique endpoint specific to the AWS account making the call.

Request syntax:

```
GET /endpoint?endpointType=endpointType
```

URI Request Parameters:

Name	Type	Req?	Description
endpointType	EndpointType	no	The endpoint type.

Response syntax:

```
Content-type: application/json

{
  "endpointAddress": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
endpointAddress	EndpointAddress	no	The endpoint. The format of the endpoint is as follows: *identifier*.iot.*region*.amazonaws.com.

Errors:

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

CLI

Synopsis:

```
aws iot  describe-endpoint \
    [--endpoint-type <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "endpointType": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
endpointType	string	The endpoint type.

Output:

```
{
  "endpointAddress": "string"
}
```

cli output fields:

Name	Type	Description
endpointAddress	string	The endpoint. The format of the endpoint is as follows: *identifier*.iot.*region*.amazonaws.com.

DescribeEventConfigurations

Describes event configurations.

Request syntax:

```
GET /event-configurations
```

Response syntax:

```
Content-type: application/json

{
```

```
  "eventConfigurations": {
    "string": {
      "Enabled": "boolean"
    }
  },
  "creationDate": "timestamp",
  "lastModifiedDate": "timestamp"
}
```

Response Body Parameters:

Name	Type	Req?	Description
eventConfigurations	EventConfigurations	no	The event configurations.
creationDate	CreationDate	no	The creation date of the event configuration.
lastModifiedDate	LastModifiedDate	no	The date the event configurations were last modified.

Errors:

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

CLI

Synopsis:

```
aws iot  describe-event-configurations  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
}
```

Output:

```
{
  "eventConfigurations": {
    "string": {
      "Enabled": "boolean"
    }
```

```
  },
  "creationDate": "timestamp",
  "lastModifiedDate": "timestamp"
}
```

cli output fields:

Name	Type	Description
eventConfigurations	map key: EventType value: Configuration	The event configurations.
EventType	string enum: THING | THING_GROUP | THING_TYPE | THING_GROUP_MEMBERSHIP | THING_GROUP_HIERARCHY | THING_TYPE_ASSOCIATION | JOB | JOB_EXECUTION java class: com.amazonaws.iot.common.types.enums.EventType	
Configuration	Configuration	
Enabled	boolean	True to enable the configuration.
creationDate	timestamp	The creation date of the event configuration.
lastModifiedDate	timestamp	The date the event configurations were last modified.

DescribeIndex

Describes a search index.

Request syntax:

```
GET /indices/indexName
```

URI Request Parameters:

Name	Type	Req?	Description
indexName	IndexName	yes	The index name.

Response syntax:

```
Content-type: application/json
```

```
{
  "indexName": "string",
  "indexStatus": "string",
  "schema": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
indexName	IndexName	no	The index name.
indexStatus	IndexStatus	no	The index status.
schema	IndexSchema	no	Contains a value that specifies the type of indexing performed. Valid values are: 1. REGISTRY – Your thing index will contain only registry data. 2. REGISTRY_AND_SHADOW - Your thing index will contain registry and shadow data.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  describe-index \
    --index-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "indexName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
indexName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The index name.

Output:

```
{
  "indexName": "string",
  "indexStatus": "string",
  "schema": "string"
}
```

cli output fields:

Name	Type	Description
indexName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The index name.
indexStatus	string enum: ACTIVE | BUILDING | REBUILDING java class: com.amazonaws.iot.indexing.IndexStatus	The index status.

Name	Type	Description
schema	string	Contains a value that specifies the type of indexing performed. Valid values are: 1. REGISTRY – Your thing index will contain only registry data. 2. REGISTRY_AND_SHADOW - Your thing index will contain registry and shadow data.

DescribeJob

Describes a job.

Request syntax:

```
GET /jobs/jobId
```

URI Request Parameters:

Name	Type	Req?	Description
jobId	JobId	yes	The unique identifier you assigned to this job when it was created.

Response syntax:

```
Content-type: application/json

{
  "documentSource": "string",
  "job": {
    "jobArn": "string",
    "jobId": "string",
    "targetSelection": "string",
    "status": "string",
    "comment": "string",
    "targets": [
      "string"
    ],
    "description": "string",
    "presignedUrlConfig": {
      "roleArn": "string",
      "expiresInSec": "long"
    },
    "jobExecutionsRolloutConfig": {
      "maximumPerMinute": "integer"
    },
    "createdAt": "timestamp",
    "lastUpdatedAt": "timestamp",
    "completedAt": "timestamp",
    "jobProcessDetails": {
```

```
      "processingTargets": [
        "string"
      ],
      "numberOfCanceledThings": "integer",
      "numberOfSucceededThings": "integer",
      "numberOfFailedThings": "integer",
      "numberOfRejectedThings": "integer",
      "numberOfQueuedThings": "integer",
      "numberOfInProgressThings": "integer",
      "numberOfRemovedThings": "integer"
    },
    "documentParameters": {
      "string": "string"
    }
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
documentSource	JobDocumentSource	no	An S3 link to the job document.
job	Job	no	Information about the job.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  describe-job \
```

```
    --job-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "jobId": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.

Output:

```
{
  "documentSource": "string",
  "job": {
    "jobArn": "string",
    "jobId": "string",
    "targetSelection": "string",
    "status": "string",
    "comment": "string",
    "targets": [
      "string"
    ],
    "description": "string",
    "presignedUrlConfig": {
      "roleArn": "string",
      "expiresInSec": "long"
    },
    "jobExecutionsRolloutConfig": {
      "maximumPerMinute": "integer"
    },
    "createdAt": "timestamp",
    "lastUpdatedAt": "timestamp",
    "completedAt": "timestamp",
    "jobProcessDetails": {
      "processingTargets": [
        "string"
      ],
      "numberOfCanceledThings": "integer",
      "numberOfSucceededThings": "integer",
      "numberOfFailedThings": "integer",
      "numberOfRejectedThings": "integer",
      "numberOfQueuedThings": "integer",
      "numberOfInProgressThings": "integer",
      "numberOfRemovedThings": "integer"
    },
    "documentParameters": {
      "string": "string"
    }
  }
}
```

cli output fields:

Name	Type	Description
documentSource	string length max:1350 min:1	An S3 link to the job document.
job	Job	Information about the job.
jobArn	string	An ARN identifying the job with format "arn:aws:iot:region:account:job/jobId".
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.
targetSelection	string enum: CONTINUOUS | SNAPSHOT java class: com.amazonaws.iot.laser.TargetSelection	Specifies whether the job will continue to run (CONTINUOUS), or will be complete after all those things specified as targets have completed the job (SNAPSHOT). If continuous, the job may also be run on a thing when a change is detected in a target. For example, a job will run on a device when the thing representing the device is added to a target group, even after the job was completed by all things originally in the group.
status	string enum: IN_PROGRESS | CANCELED | COMPLETED java class: ccom.amazonaws.iot.laser.common.JobStatus	The status of the job, one of `IN_PROGRESS`, `CANCELED`, or `COMPLETED`.
comment	string length max:2028 pattern: [^\\p{C}]+	If the job was updated, describes the reason for the update.
targets	list member: TargetArn	A list of IoT things and thing groups to which the job should be sent.
TargetArn	string	
description	string length max:2028	A short text description of the job.

Name	Type	Description
	pattern: [^\\p{C}]+	
presignedUrlConfig	PresignedUrlConfig	Configuration for pre-signed S3 URLs.
roleArn	string length max:2048 min:20	The ARN of an IAM role that grants grants permission to download files from the S3 bucket where the job data/ updates are stored. The role must also grant permission for IoT to download the files.
expiresInSec	long java class: java.lang.Long range- max:3600 min:60	How long (in seconds) pre-signed URLs are valid. Valid values are 60 - 3600, the default value is 3600 seconds. Pre-signed URLs are generated when Jobs receives an MQTT request for the job document.
jobExecutionsRolloutConfig	JobExecutionsRolloutConfig	Allows you to create a staged rollout of a job.
maximumPerMinute	integer java class: java.lang.Integer range- max:1000 min:1	The maximum number of things that will be notified of a pending job, per minute. This parameter allows you to create a staged rollout.
createdAt	timestamp	The time, in milliseconds since the epoch, when the job was created.
lastUpdatedAt	timestamp	The time, in milliseconds since the epoch, when the job was last updated.
completedAt	timestamp	The time, in milliseconds since the epoch, when the job was completed.
jobProcessDetails	JobProcessDetails	Details about the job process.
processingTargets	list member: ProcessingTargetName java class: java.util.List	The devices on which the job is executing.
ProcessingTargetName	string	
numberOfCanceledThings	integer java class: java.lang.Integer	The number of things that cancelled the job.

Name	Type	Description
numberOfSucceededThings	integer java class: java.lang.Integer	The number of things which successfully completed the job.
numberOfFailedThings	integer java class: java.lang.Integer	The number of things that failed executing the job.
numberOfRejectedThings	integer java class: java.lang.Integer	The number of things that rejected the job.
numberOfQueuedThings	integer java class: java.lang.Integer	The number of things that are awaiting execution of the job.
numberOfInProgressThings	integer java class: java.lang.Integer	The number of things currently executing the job.
numberOfRemovedThings	integer java class: java.lang.Integer	The number of things that are no longer scheduled to execute the job because they have been deleted or have been removed from the group that was a target of the job.
documentParameters	map key: ParameterKey value: ParameterValue	The parameters specified for the job document.
ParameterKey	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
ParameterValue	string length max:1024 min:1 pattern: [^\\p{C}]+	

DescribeJobExecution

Describes a job execution.

Request syntax:

```
GET /things/thingName/jobs/jobId?executionNumber=executionNumber
```

URI Request Parameters:

Name	Type	Req?	Description
jobId	JobId	yes	The unique identifier you assigned to this job when it was created.
thingName	ThingName	yes	The name of the thing on which the job execution is running.
executionNumber	ExecutionNumber	no	A string (consisting of the digits "0" through "9" which is used to specify a particular job execution on a particular device.

Response syntax:

```
Content-type: application/json

{
  "execution": {
    "jobId": "string",
    "status": "string",
    "statusDetails": {
      "detailsMap": {
        "string": "string"
      }
    },
    "thingArn": "string",
    "queuedAt": "timestamp",
    "startedAt": "timestamp",
    "lastUpdatedAt": "timestamp",
    "executionNumber": "long"
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
execution	JobExecution	no	Information about the job execution.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  describe-job-execution \
    --job-id <value> \
    --thing-name <value> \
    [--execution-number <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "jobId": "string",
  "thingName": "string",
  "executionNumber": "long"
}
```

`cli-input-json` fields:

Name	Type	Description
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing on which the job execution is running.
executionNumber	long java class: java.lang.Long	A string (consisting of the digits "0" through "9" which is used to specify a particular job execution on a particular device.

Output:

```
{
  "execution": {
    "jobId": "string",
    "status": "string",
    "statusDetails": {
      "detailsMap": {
        "string": "string"
      }
    },
    "thingArn": "string",
    "queuedAt": "timestamp",
    "startedAt": "timestamp",
    "lastUpdatedAt": "timestamp",
    "executionNumber": "long"
  }
}
```

cli output fields:

Name	Type	Description
execution	JobExecution	Information about the job execution.
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to the job when it was created.
status	string enum: QUEUED | IN_PROGRESS | SUCCEEDED | FAILED | REJECTED | REMOVED | CANCELED java class: com.amazonaws.iot.laser.common.JobExecutionStatus	The status of the job execution (IN_PROGRESS, QUEUED, FAILED, SUCCESS, CANCELED, or REJECTED).
statusDetails	JobExecutionStatusDetails	A collection of name/value pairs that describe the status of the job execution.
detailsMap	map key: DetailsKey value: DetailsValue	The job execution status.
DetailsKey	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
DetailsValue	string length max:1024 min:1 pattern: [^\\p{C}]*+	

Name	Type	Description
thingArn	string	The ARN of the thing on which the job execution is running.
queuedAt	timestamp	The time, in milliseconds since the epoch, when the job execution was queued.
startedAt	timestamp	The time, in milliseconds since the epoch, when the job execution started.
lastUpdatedAt	timestamp	The time, in milliseconds since the epoch, when the job execution was last updated.
executionNumber	long java class: java.lang.Long	A string (consisting of the digits "0" through "9") which identifies this particular job execution on this particular device. It can be used in commands which return or update job execution information.

DescribeJobExecution

Gets details of a job execution.

Request syntax:

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument
```

URI Request Parameters:

Name	Type	Req?	Description
jobId	DescribeJobExecutionJobId	yes	The unique identifier assigned to this job when it was created.
thingName	ThingName	yes	The thing name associated with the device the job execution is running on.
includeJobDocument	IncludeJobDocument	no	Optional. When set to true, the response contains the job document. The default is false.
executionNumber	ExecutionNumber	no	Optional. A number that identifies a particular job execution

Name	Type	Req?	Description
			on a particular device. If not specified, the latest job execution is returned.

Response syntax:

```
Content-type: application/json

{
  "execution": {
    "jobId": "string",
    "thingName": "string",
    "status": "string",
    "statusDetails": {
      "string": "string"
    },
    "queuedAt": "long",
    "startedAt": "long",
    "lastUpdatedAt": "long",
    "versionNumber": "long",
    "executionNumber": "long",
    "jobDocument": "string"
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
execution	JobExecution	no	Contains data about a job execution.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`CertificateValidationException`

The certificate is invalid.

HTTP response code: 400

`TerminalStateException`

The job is in a terminal state.

HTTP response code: 410

CLI

Synopsis:

```
aws iot  describe-job-execution \
    --job-id <value> \
    --thing-name <value> \
    [--include-job-document | --no-include-job-document] \
    [--execution-number <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "jobId": "string",
  "thingName": "string",
  "includeJobDocument": "boolean",
  "executionNumber": "long"
}
```

`cli-input-json` fields:

Name	Type	Description
jobId	string pattern: [a-zA-Z0-9_-]+|^$next	The unique identifier assigned to this job when it was created.
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The thing name associated with the device the job execution is running on.
includeJobDocument	boolean java class: java.lang.Boolean	Optional. When set to true, the response contains the job document. The default is false.
executionNumber	long java class: java.lang.Long	Optional. A number that identifies a particular job execution on a particular device. If not specified, the latest job execution is returned.

Output:

```
{
  "execution": {
    "jobId": "string",
    "thingName": "string",
    "status": "string",
    "statusDetails": {
      "string": "string"
    },
    "queuedAt": "long",
    "startedAt": "long",
    "lastUpdatedAt": "long",
    "versionNumber": "long",
    "executionNumber": "long",
    "jobDocument": "string"
  }
}
```

cli output fields:

Name	Type	Description
execution	JobExecution	Contains data about a job execution.
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing that is executing the job.
status	string enum: QUEUED | IN_PROGRESS | SUCCEEDED | FAILED | REJECTED | REMOVED | CANCELED java class: com.amazonaws.iot.laser.common.JobExecutionStatus	The status of the job execution. Can be one of: "QUEUED", "IN_PROGRESS", "FAILED", "SUCCESS", "CANCELED", "REJECTED", or "REMOVED".
statusDetails	map key: DetailsKey value: DetailsValue	A collection of name/value pairs that describe the status of the job execution.
DetailsKey	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
DetailsValue	string length max:1024 min:1	

Name	Type	Description
	pattern: [^\\p{C}]*+	
queuedAt	long	The time, in milliseconds since the epoch, when the job execution was enqueued.
startedAt	long java class: java.lang.Long	The time, in milliseconds since the epoch, when the job execution was started.
lastUpdatedAt	long	The time, in milliseconds since the epoch, when the job execution was last updated.
versionNumber	long	The version of the job execution. Job execution versions are incremented each time they are updated by a device.
executionNumber	long java class: java.lang.Long	A number that identifies a particular job execution on a particular device. It can be used later in commands that return or update job execution information.
jobDocument	string length max:32768	The content of the job document.

DescribeRoleAlias

Describes a role alias.

Request syntax:

```
GET /role-aliases/roleAlias
```

URI Request Parameters:

Name	Type	Req?	Description
roleAlias	RoleAlias	yes	The role alias to describe.

Response syntax:

```
Content-type: application/json

{
  "roleAliasDescription": {
    "roleAlias": "string",
    "roleArn": "string",
```

```
    "owner": "string",
    "credentialDurationSeconds": "integer",
    "creationDate": "timestamp",
    "lastModifiedDate": "timestamp"
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
roleAliasDescription	RoleAliasDescription	no	The role alias description.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  describe-role-alias \
```

```
    --role-alias <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "roleAlias": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
roleAlias	string length max:128 min:1 pattern: [w=,@-]+	The role alias to describe.

Output:

```
{
  "roleAliasDescription": {
    "roleAlias": "string",
    "roleArn": "string",
    "owner": "string",
    "credentialDurationSeconds": "integer",
    "creationDate": "timestamp",
    "lastModifiedDate": "timestamp"
  }
}
```

cli output fields:

Name	Type	Description
roleAliasDescription	RoleAliasDescription	The role alias description.
roleAlias	string length max:128 min:1 pattern: [w=,@-]+	The role alias.
roleArn	string length max:2048 min:20	The role ARN.
owner	string pattern: [0-9]{12}	The role alias owner.
credentialDurationSeconds	integer java class: java.lang.Integer range- max:3600 min:900	The number of seconds for which the credential is valid.

Name	Type	Description
creationDate	timestamp	The UNIX timestamp of when the role alias was created.
lastModifiedDate	timestamp	The UNIX timestamp of when the role alias was last modified.

DescribeStream

Gets information about a stream.

Request syntax:

```
GET /streams/streamId
```

URI Request Parameters:

Name	Type	Req?	Description
streamId	StreamId	yes	The stream ID.

Response syntax:

```
Content-type: application/json

{
  "streamInfo": {
    "streamId": "string",
    "streamArn": "string",
    "streamVersion": "integer",
    "description": "string",
    "files": [
      {
        "fileId": "integer",
        "s3Location": {
          "bucket": "string",
          "key": "string",
          "version": "string"
        }
      }
    ],
    "createdAt": "timestamp",
    "lastUpdatedAt": "timestamp",
    "roleArn": "string"
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
streamInfo	StreamInfo	no	Information about the stream.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  describe-stream \
    --stream-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "streamId": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
streamId	string length max:128 min:1	The stream ID.

Name	Type	Description
	pattern: [a-zA-Z0-9_-]+	

Output:

```
{
  "streamInfo": {
    "streamId": "string",
    "streamArn": "string",
    "streamVersion": "integer",
    "description": "string",
    "files": [
      {
        "fileId": "integer",
        "s3Location": {
          "bucket": "string",
          "key": "string",
          "version": "string"
        }
      }
    ],
    "createdAt": "timestamp",
    "lastUpdatedAt": "timestamp",
    "roleArn": "string"
  }
}
```

cli output fields:

Name	Type	Description
streamInfo	StreamInfo	Information about the stream.
streamId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The stream ID.
streamArn	string	The stream ARN.
streamVersion	integer java class: java.lang.Integer range- max:65535 min:0	The stream version.
description	string length max:2028 pattern: [^\\p{C}]+	The description of the stream.
files	list member: StreamFile	The files to stream.
StreamFile	StreamFile	
fileId	integer	The file ID.

Name	Type	Description
	java class: java.lang.Integer range- max:255 min:0	
s3Location	S3Location	The location of the file in S3.
bucket	string length min:1	The S3 bucket that contains the file to stream.
key	string length min:1	The name of the file within the S3 bucket to stream.
version	string	The file version.
createdAt	timestamp	The date when the stream was created.
lastUpdatedAt	timestamp	The date when the stream was last updated.
roleArn	string length max:2048 min:20	An IAM role AWS IoT assumes to access your S3 files.

DescribeThing

Gets information about the specified thing.

Request syntax:

```
GET /things/thingName
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	yes	The name of the thing.

Response syntax:

```
Content-type: application/json

{
  "defaultClientId": "string",
  "thingName": "string",
  "thingId": "string",
  "thingArn": "string",
  "thingTypeName": "string",
  "attributes": {
    "string": "string"
  },
  "version": "long"
```

```
}
```

Response Body Parameters:

Name	Type	Req?	Description
defaultClientId	ClientId	no	The default client ID.
thingName	ThingName	no	The name of the thing.
thingId	ThingId	no	The ID of the thing to describe.
thingArn	ThingArn	no	The ARN of the thing to describe.
thingTypeName	ThingTypeName	no	The thing type name.
attributes	Attributes	no	The thing attributes.
version	Version	no	The current version of the thing record in the registry. **Note** To avoid unintentional changes to the information in the registry, you can pass the version information in the `expectedVersion` parameter of the `UpdateThing` and `DeleteThing` calls.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  describe-thing \
    --thing-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing.

Output:

```
{
  "defaultClientId": "string",
  "thingName": "string",
  "thingId": "string",
  "thingArn": "string",
  "thingTypeName": "string",
  "attributes": {
    "string": "string"
  },
  "version": "long"
}
```

cli output fields:

Name	Type	Description
defaultClientId	string	The default client ID.
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing.
thingId	string	The ID of the thing to describe.
thingArn	string	The ARN of the thing to describe.
thingTypeName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The thing type name.
attributes	map key: AttributeName value: AttributeValue	The thing attributes.
AttributeName	string length max:128 pattern: [a-zA-Z0-9_.,@/:#-]+	
AttributeValue	string length max:800 pattern: [a-zA-Z0-9_.,@/:#-]*	
version	long	The current version of the thing record in the registry. **Note** To avoid unintentional changes to the information in the registry, you can pass the version information in the `expectedVersion` parameter of the `UpdateThing` and `DeleteThing` calls.

DescribeThingGroup

Describe a thing group.

Request syntax:

```
GET /thing-groups/thingGroupName
```

URI Request Parameters:

Name	Type	Req?	Description
thingGroupName	ThingGroupName	yes	The name of the thing group.

Response syntax:

```
Content-type: application/json

{
  "thingGroupName": "string",
  "thingGroupId": "string",
  "thingGroupArn": "string",
  "version": "long",
  "thingGroupProperties": {
    "thingGroupDescription": "string",
    "attributePayload": {
      "attributes": {
        "string": "string"
      },
      "merge": "boolean"
    }
  },
  "thingGroupMetadata": {
    "parentGroupName": "string",
    "rootToParentThingGroups": [
      {
        "groupName": "string",
        "groupArn": "string"
      }
    ],
    "creationDate": "timestamp"
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
thingGroupName	ThingGroupName	no	The name of the thing group.
thingGroupId	ThingGroupId	no	The thing group ID.
thingGroupArn	ThingGroupArn	no	The thing group ARN.
version	Version	no	The version of the thing group.
thingGroupProperties	ThingGroupProperties	no	The thing group properties.
thingGroupMetadata	ThingGroupMetadata	no	Thing group metadata.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  describe-thing-group \
    --thing-group-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingGroupName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
thingGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing group.

Output:

```
{
  "thingGroupName": "string",
  "thingGroupId": "string",
```

```
  "thingGroupArn": "string",
  "version": "long",
  "thingGroupProperties": {
    "thingGroupDescription": "string",
    "attributePayload": {
      "attributes": {
        "string": "string"
      },
      "merge": "boolean"
    }
  },
  "thingGroupMetadata": {
    "parentGroupName": "string",
    "rootToParentThingGroups": [
      {
        "groupName": "string",
        "groupArn": "string"
      }
    ],
    "creationDate": "timestamp"
  }
}
```

cli output fields:

Name	Type	Description
thingGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing group.
thingGroupId	string length max:128 min:1 pattern: [a-zA-Z0-9-]+	The thing group ID.
thingGroupArn	string	The thing group ARN.
version	long	The version of the thing group.
thingGroupProperties	ThingGroupProperties	The thing group properties.
thingGroupDescription	string length max:2028 pattern: [\\p{Graph}]*	The thing group description.
attributePayload	AttributePayload	The thing group attributes in JSON format.
attributes	map key: AttributeName value: AttributeValue	A JSON string containing up to three key-value pair in JSON format. For example: *\"attributes\": {\"string1\": \"string2\"}*
AttributeName	string	

Name	Type	Description
	length max:128 pattern: [a-zA-Z0-9_.,@/:#-]+	
AttributeValue	string length max:800 pattern: [a-zA-Z0-9_.,@/:#-]*	
merge	boolean	Specifies whether the list of attributes provided in the `AttributePayload` is merged with the attributes stored in the registry, instead of overwriting them. To remove an attribute, call `UpdateThing` with an empty attribute value. **Note** The `merge` attribute is only valid when calling `UpdateThing`.
thingGroupMetadata	ThingGroupMetadata	Thing group metadata.
parentGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The parent thing group name.
rootToParentThingGroups	list member: GroupNameAndArn java class: java.util.List	The root parent thing group.
GroupNameAndArn	GroupNameAndArn	
groupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The group name.
groupArn	string	The group ARN.
creationDate	timestamp	The UNIX timestamp of when the thing group was created.

DescribeThingRegistrationTask

Describes a bulk thing provisioning task.

Request syntax:

```
GET /thing-registration-tasks/taskId
```

URI Request Parameters:

Name	Type	Req?	Description
taskId	TaskId	yes	The task ID.

Response syntax:

```
Content-type: application/json

{
  "taskId": "string",
  "creationDate": "timestamp",
  "lastModifiedDate": "timestamp",
  "templateBody": "string",
  "inputFileBucket": "string",
  "inputFileKey": "string",
  "roleArn": "string",
  "status": "string",
  "message": "string",
  "successCount": "integer",
  "failureCount": "integer",
  "percentageProgress": "integer"
}
```

Response Body Parameters:

Name	Type	Req?	Description
taskId	TaskId	no	The task ID.
creationDate	CreationDate	no	The task creation date.
lastModifiedDate	LastModifiedDate	no	The date when the task was last modified.
templateBody	TemplateBody	no	The task's template.
inputFileBucket	RegistryS3BucketName	no	The S3 bucket that contains the input file.
inputFileKey	RegistryS3KeyName	no	The input file key.
roleArn	RoleArn	no	The role ARN that grants access to the input file bucket.
status	Status	no	The status of the bulk thing provisioning task.
message	ErrorMessage	no	The message.
successCount	Count	no	The number of things successfully provisioned.

Name	Type	Req?	Description
failureCount	Count	no	The number of things that failed to be provisioned.
percentageProgress	Percentage	no	The progress of the bulk provisioning task expressed as a percentage.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  describe-thing-registration-task \
    --task-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "taskId": "string"
```

```
}
```

`cli-input-json` fields:

Name	Type	Description
taskId	string length max:40	The task ID.

Output:

```
{
  "taskId": "string",
  "creationDate": "timestamp",
  "lastModifiedDate": "timestamp",
  "templateBody": "string",
  "inputFileBucket": "string",
  "inputFileKey": "string",
  "roleArn": "string",
  "status": "string",
  "message": "string",
  "successCount": "integer",
  "failureCount": "integer",
  "percentageProgress": "integer"
}
```

cli output fields:

Name	Type	Description
taskId	string length max:40	The task ID.
creationDate	timestamp	The task creation date.
lastModifiedDate	timestamp	The date when the task was last modified.
templateBody	string	The task's template.
inputFileBucket	string length max:256 min:3 pattern: [a-zA-Z0-9._-]+	The S3 bucket that contains the input file.
inputFileKey	string length max:1024 min:1 pattern: [a-zA-Z0-9!_.*'()-/]+	The input file key.
roleArn	string length max:2048 min:20	The role ARN that grants access to the input file bucket.
status	string	The status of the bulk thing provisioning task.

Name	Type	Description
	enum: InProgress | Completed | Failed | Cancelled | Cancelling java class: com.amazonaws.iot.common.types.enums.Status	
message	string length max:2048	The message.
successCount	integer	The number of things successfully provisioned.
failureCount	integer	The number of things that failed to be provisioned.
percentageProgress	integer range- max:100 min:0	The progress of the bulk provisioning task expressed as a percentage.

DescribeThingType

Gets information about the specified thing type.

Request syntax:

```
GET /thing-types/thingTypeName
```

URI Request Parameters:

Name	Type	Req?	Description
thingTypeName	ThingTypeName	yes	The name of the thing type.

Response syntax:

```
Content-type: application/json

{
  "thingTypeName": "string",
  "thingTypeId": "string",
  "thingTypeArn": "string",
  "thingTypeProperties": {
    "thingTypeDescription": "string",
    "searchableAttributes": [
      "string"
    ]
  },
  "thingTypeMetadata": {
    "deprecated": "boolean",
    "deprecationDate": "timestamp",
    "creationDate": "timestamp"
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
thingTypeName	ThingTypeName	no	The name of the thing type.
thingTypeId	ThingTypeId	no	The thing type ID.
thingTypeArn	ThingTypeArn	no	The thing type ARN.
thingTypeProperties	ThingTypeProperties	no	The ThingTypeProperties contains information about the thing type including description, and a list of searchable thing attribute names.
thingTypeMetadata	ThingTypeMetadata	no	The ThingTypeMetadata contains additional information about the thing type including: creation date and time, a value indicating whether the thing type is deprecated, and a date and time when it was deprecated.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  describe-thing-type \
    --thing-type-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingTypeName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
thingTypeName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing type.

Output:

```
{
  "thingTypeName": "string",
  "thingTypeId": "string",
  "thingTypeArn": "string",
  "thingTypeProperties": {
    "thingTypeDescription": "string",
    "searchableAttributes": [
      "string"
    ]
  },
  "thingTypeMetadata": {
    "deprecated": "boolean",
    "deprecationDate": "timestamp",
    "creationDate": "timestamp"
  }
}
```

cli output fields:

Name	Type	Description
thingTypeName	string	The name of the thing type.

Name	Type	Description
	length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
thingTypeId	string	The thing type ID.
thingTypeArn	string	The thing type ARN.
thingTypeProperties	ThingTypeProperties	The ThingTypeProperties contains information about the thing type including description, and a list of searchable thing attribute names.
thingTypeDescription	string length max:2028 pattern: [\\p{Graph}]*	The description of the thing type.
searchableAttributes	list member: AttributeName java class: java.util.List	A list of searchable thing attribute names.
AttributeName	string length max:128 pattern: [a-zA-Z0-9_.,@/:#-]+	
thingTypeMetadata	ThingTypeMetadata	The ThingTypeMetadata contains additional information about the thing type including: creation date and time, a value indicating whether the thing type is deprecated, and a date and time when it was deprecated.
deprecated	boolean	Whether the thing type is deprecated. If **true**, no new things could be associated with this type.
deprecationDate	timestamp	The date and time when the thing type was deprecated.
creationDate	timestamp	The date and time when the thing type was created.

DetachPolicy

Detaches a policy from the specified target.

Request syntax:

```
POST /target-policies/policyName
Content-type: application/json

{
  "target": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
policyName	PolicyName	yes	The policy to detach.

Request Body Parameters:

Name	Type	Req?	Description
target	PolicyTarget	yes	The target from which the policy will be detached.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`LimitExceededException`

The number of attached entities exceeds the limit.

HTTP response code: 410

CLI

Synopsis:

```
aws iot  detach-policy \
    --policy-name <value> \
    --target <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "policyName": "string",
  "target": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy to detach.
target	string	The target from which the policy will be detached.

Output:

None

DetachPrincipalPolicy

Removes the specified policy from the specified certificate.

Note: This API is deprecated. Please use DetachPolicy instead.

Request syntax:

```
DELETE /principal-policies/policyName
x-amzn-iot-principal: principal
```

URI Request Parameters:

Name	Type	Req?	Description
policyName	PolicyName	yes	The name of the policy to detach.

Name	Type	Req?	Description
principal	Principal	yes	The principal. If the principal is a certificate, specify the certificate ARN. If the principal is an Amazon Cognito identity, specify the identity ID.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  detach-principal-policy \
    --policy-name <value> \
    --principal <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "policyName": "string",
  "principal": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The name of the policy to detach.
principal	string	The principal. If the principal is a certificate, specify the certificate ARN. If the principal is an Amazon Cognito identity, specify the identity ID.

Output:

None

DetachThingPrincipal

Detaches the specified principal from the specified thing.

Request syntax:

```
DELETE /things/thingName/principals
x-amzn-principal: principal
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	yes	The name of the thing.
principal	Principal	yes	If the principal is a certificate, this value must be ARN of the certificate. If the principal is an Amazon Cognito identity, this value must be the ID of the Amazon Cognito identity.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  detach-thing-principal \
    --thing-name <value> \
    --principal <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string",
  "principal": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing.

Name	Type	Description
principal	string	If the principal is a certificate, this value must be ARN of the certificate. If the principal is an Amazon Cognito identity, this value must be the ID of the Amazon Cognito identity.

Output:

None

DisableTopicRule

Disables the rule.

Request syntax:

```
POST /rules/ruleName/disable
```

URI Request Parameters:

Name	Type	Req?	Description
ruleName	RuleName	yes	The name of the rule to disable.

Errors:

`InternalException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

CLI

Synopsis:

```
aws iot  disable-topic-rule \
    --rule-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "ruleName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
ruleName	string length max:128 min:1 pattern: ^[a-zA-Z0-9_]+$	The name of the rule to disable.

Output:

None

EnableTopicRule

Enables the rule.

Request syntax:

```
POST /rules/ruleName/enable
```

URI Request Parameters:

Name	Type	Req?	Description
ruleName	RuleName	yes	The name of the topic rule to enable.

Errors:

`InternalException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

CLI

Synopsis:

```
aws iot  enable-topic-rule \
    --rule-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "ruleName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
ruleName	string length max:128 min:1 pattern: ^[a-zA-Z0-9_]+$	The name of the topic rule to enable.

Output:

None

GetEffectivePolicies

Gets effective policies.

Request syntax:

```
POST /effective-policies?thingName=thingName
Content-type: application/json
```

```
{
  "principal": "string",
  "cognitoIdentityPoolId": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	no	The thing name.

Request Body Parameters:

Name	Type	Req?	Description
principal	Principal	no	The principal.
cognitoIdentityPoolId	CognitoIdentityPoolId	no	The Cognito identity pool ID.

Response syntax:

```
Content-type: application/json

{
  "effectivePolicies": [
    {
      "policyName": "string",
      "policyArn": "string",
      "policyDocument": "string"
    }
  ]
}
```

Response Body Parameters:

Name	Type	Req?	Description
effectivePolicies	EffectivePolicies	no	The effective policies.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`LimitExceededException`

The number of attached entities exceeds the limit.

HTTP response code: 410

CLI

Synopsis:

```
aws iot  get-effective-policies \
    [--principal <value>] \
    [--cognito-identity-pool-id <value>] \
    [--thing-name <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "principal": "string",
  "cognitoIdentityPoolId": "string",
  "thingName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
principal	string	The principal.
cognitoIdentityPoolId	string	The Cognito identity pool ID.
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The thing name.

Output:

```
{
```

```
  "effectivePolicies": [
    {
      "policyName": "string",
      "policyArn": "string",
      "policyDocument": "string"
    }
  ]
}
```

cli output fields:

Name	Type	Description
effectivePolicies	list member: EffectivePolicy java class: java.util.List	The effective policies.
EffectivePolicy	EffectivePolicy	
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.
policyArn	string	The policy ARN.
policyDocument	string	The IAM policy document.

GetIndexingConfiguration

Gets the search configuration.

Request syntax:

```
GET /indexing/config
```

Response syntax:

```
Content-type: application/json

{
  "thingIndexingConfiguration": {
    "thingIndexingMode": "string"
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
thingIndexingConfiguration	ThingIndexingConfiguration	no	Thing indexing configuration.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  get-indexing-configuration  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
}
```

Output:

```
{
  "thingIndexingConfiguration": {
    "thingIndexingMode": "string"
  }
}
```

cli output fields:

Name	Type	Description
thingIndexingConfiguration	ThingIndexingConfiguration	Thing indexing configuration.
thingIndexingMode	string	Thing indexing mode. Valid values are:

Name	Type	Description
	enum: OFF | REGISTRY | REGISTRY_AND_SHADOW java class: com.amazonaws.iot.indexing.ThingIndexingMode	• REGISTRY – Your thing index will contain only registry data. • REGISTRY_AND_SHADOW - Your thing index will contain registry and shadow data. • OFF - Thing indexing is disabled.

GetJobDocument

Gets a job document.

Request syntax:

```
GET /jobs/jobId/job-document
```

URI Request Parameters:

Name	Type	Req?	Description
jobId	JobId	yes	The unique identifier you assigned to this job when it was created.

Response syntax:

```
Content-type: application/json

{
  "document": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
document	JobDocument	no	The job document content.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  get-job-document \
    --job-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "jobId": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.

Output:

```
{
  "document": "string"
}
```

cli output fields:

Name	Type	Description
document	string length max:32768	The job document content.

GetLoggingOptions

Gets the logging options.

Request syntax:

```
GET /loggingOptions
```

Response syntax:

```
Content-type: application/json

{
  "roleArn": "string",
  "logLevel": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
roleArn	AwsArn	no	The ARN of the IAM role that grants access.
logLevel	LogLevel	no	The logging level.

Errors:

`InternalException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  get-logging-options  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
}
```

Output:

```
{
  "roleArn": "string",
  "logLevel": "string"
}
```

cli output fields:

Name	Type	Description
roleArn	string	The ARN of the IAM role that grants access.
logLevel	string enum: DEBUG | INFO | ERROR | WARN | DISABLED java class: iot.goldeneye.service.LogLevel	The logging level.

GetOTAUpdate

Gets an OTA update.

Request syntax:

```
GET /otaUpdates/otaUpdateId
```

URI Request Parameters:

Name	Type	Req?	Description
otaUpdateId	OTAUpdateId	yes	The OTA update ID.

Response syntax:

```
Content-type: application/json

{
  "otaUpdateInfo": {
    "otaUpdateId": "string",
    "otaUpdateArn": "string",
    "creationDate": "timestamp",
    "lastModifiedDate": "timestamp",
    "description": "string",
    "targets": [
      "string"
    ],
    "targetSelection": "string",
    "otaUpdateFiles": [
      {
        "fileName": "string",
        "fileVersion": "string",
```

```
        "fileSource": {
          "streamId": "string",
          "fileId": "integer"
        },
        "codeSigning": {
          "awsSignerJobId": "string",
          "customCodeSigning": {
            "signature": {
              "stream": {
                "streamId": "string",
                "fileId": "integer"
              },
              "inlineDocument": "blob"
            },
            "certificateChain": {
              "stream": {
                "streamId": "string",
                "fileId": "integer"
              },
              "certificateName": "string",
              "inlineDocument": "string"
            },
            "hashAlgorithm": "string",
            "signatureAlgorithm": "string"
          }
        },
        "attributes": {
          "string": "string"
        }
      }
    ],
    "otaUpdateStatus": "string",
    "awsIotJobId": "string",
    "awsIotJobArn": "string",
    "errorInfo": {
      "code": "string",
      "message": "string"
    },
    "additionalParameters": {
      "string": "string"
    }
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
otaUpdateInfo	OTAUpdateInfo	no	The OTA update info.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  get-ota-update \
    --ota-update-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "otaUpdateId": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
otaUpdateId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The OTA update ID.

Output:

```
{
  "otaUpdateInfo": {
    "otaUpdateId": "string",
    "otaUpdateArn": "string",
    "creationDate": "timestamp",
    "lastModifiedDate": "timestamp",
    "description": "string",
    "targets": [
```

```
      "string"
    ],
    "targetSelection": "string",
    "otaUpdateFiles": [
      {
        "fileName": "string",
        "fileVersion": "string",
        "fileSource": {
          "streamId": "string",
          "fileId": "integer"
        },
        "codeSigning": {
          "awsSignerJobId": "string",
          "customCodeSigning": {
            "signature": {
              "stream": {
                "streamId": "string",
                "fileId": "integer"
              },
              "inlineDocument": "blob"
            },
            "certificateChain": {
              "stream": {
                "streamId": "string",
                "fileId": "integer"
              },
              "certificateName": "string",
              "inlineDocument": "string"
            },
            "hashAlgorithm": "string",
            "signatureAlgorithm": "string"
          }
        },
        "attributes": {
          "string": "string"
        }
      }
    ],
    "otaUpdateStatus": "string",
    "awsIotJobId": "string",
    "awsIotJobArn": "string",
    "errorInfo": {
      "code": "string",
      "message": "string"
    },
    "additionalParameters": {
      "string": "string"
    }
  }
}
```

cli output fields:

Name	Type	Description
otaUpdateInfo	OTAUpdateInfo	The OTA update info.
otaUpdateId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The OTA update ID.
otaUpdateArn	string	The OTA update ARN.

Name	Type	Description
creationDate	timestamp	The date when the OTA update was created.
lastModifiedDate	timestamp	The date when the OTA update was last updated.
description	string length max:2028 pattern: [^\\p{C}]+	A description of the OTA update.
targets	list member: Target	The targets of the OTA update.
Target	string	
targetSelection	string enum: CONTINUOUS | SNAPSHOT	Specifies whether the OTA update will continue to run (CONTINUOUS), or will be complete after all those things specified as targets have completed the OTA update (SNAPSHOT). If continuous, the OTA update may also be run on a thing when a change is detected in a target. For example, an OTA update will run on a thing when the thing is added to a target group, even after the OTA update was completed by all things originally in the group.
otaUpdateFiles	list member: OTAUpdateFile	A list of files associated with the OTA update.
OTAUpdateFile	OTAUpdateFile	
fileName	string	The name of the file.
fileVersion	string	The file version.
fileSource	Stream	The source of the file.
streamId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The stream ID.
fileId	integer java class: java.lang.Integer range- max:255 min:0	The ID of a file associated with a stream.

Name	Type	Description
codeSigning	CodeSigning	The code signing method of the file.
awsSignerJobId	string	The ID of the AWSSignerJob which was created to sign the file.
customCodeSigning	CustomCodeSigning	A custom method for code signing a file.
signature	CodeSigningSignature	The signature for the file.
stream	Stream	A stream of the code signing signature.
streamId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The stream ID.
fileId	integer java class: java.lang.Integer range- max:255 min:0	The ID of a file associated with a stream.
inlineDocument	blob	A base64 encoded binary representation of the code signing signature.
certificateChain	CodeSigningCertificateChain	The certificate chain.
stream	Stream	A stream of the certificate chain files.
streamId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The stream ID.
fileId	integer java class: java.lang.Integer range- max:255 min:0	The ID of a file associated with a stream.
certificateName	string	The name of the certificate.
inlineDocument	string	A base64 encoded binary representation of the code signing certificate chain.
hashAlgorithm	string	The hash algorithm used to code sign the file.
signatureAlgorithm	string	The signature algorithm used to code sign the file.

Name	Type	Description
attributes	map key: Key value: Value	A list of name/attribute pairs.
Key	string	
Value	string	
otaUpdateStatus	string enum: CREATE_PENDING | CREATE_IN_PROGRESS | CREATE_COMPLETE | CREATE_FAILED	The status of the OTA update.
awsIotJobId	string	The AWS IoT job ID associated with the OTA update.
awsIotJobArn	string	The AWS IoT job ARN associated with the OTA update.
errorInfo	ErrorInfo	Error information associated with the OTA update.
code	string	The error code.
message	string	The error message.
additionalParameters	map key: Key value: Value	A collection of name/value pairs
Key	string	
Value	string	

GetPendingJobExecutions

Gets the list of all jobs for a thing that are not in a terminal status.

Request syntax:

```
GET /things/thingName/jobs
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	yes	The name of the thing that is executing the job.

Response syntax:

```
Content-type: application/json

{
  "inProgressJobs": [
    {
      "jobId": "string",
      "queuedAt": "long",
      "startedAt": "long",
      "lastUpdatedAt": "long",
      "versionNumber": "long",
      "executionNumber": "long"
    }
  ],
  "queuedJobs": [
    {
      "jobId": "string",
      "queuedAt": "long",
      "startedAt": "long",
      "lastUpdatedAt": "long",
      "versionNumber": "long",
      "executionNumber": "long"
    }
  ]
}
```

Response Body Parameters:

Name	Type	Req?	Description
inProgressJobs	JobExecutionSummaryList	no	A list of JobExecutionSummary objects with status IN_PROGRESS.
queuedJobs	JobExecutionSummaryList	no	A list of JobExecutionSummary objects with status QUEUED.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`CertificateValidationException`

The certificate is invalid.

HTTP response code: 400

CLI

Synopsis:

```
aws iot  get-pending-job-executions \
    --thing-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing that is executing the job.

Output:

```
{
  "inProgressJobs": [
    {
      "jobId": "string",
      "queuedAt": "long",
      "startedAt": "long",
      "lastUpdatedAt": "long",
      "versionNumber": "long",
      "executionNumber": "long"
    }
  ],
  "queuedJobs": [
    {
      "jobId": "string",
      "queuedAt": "long",
      "startedAt": "long",
      "lastUpdatedAt": "long",
      "versionNumber": "long",
      "executionNumber": "long"
```

```
    }
  ]
}
```

cli output fields:

Name	Type	Description
inProgressJobs	list member: JobExecutionSummary java class: java.util.List	A list of JobExecutionSummary objects with status IN_PROGRESS.
JobExecutionSummary	JobExecutionSummary	
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.
queuedAt	long	The time, in milliseconds since the epoch, when the job execution was enqueued.
startedAt	long java class: java.lang.Long	The time, in milliseconds since the epoch, when the job execution started.
lastUpdatedAt	long	The time, in milliseconds since the epoch, when the job execution was last updated.
versionNumber	long	The version of the job execution. Job execution versions are incremented each time AWS IoT Jobs receives an update from a device.
executionNumber	long java class: java.lang.Long	A number that identifies a particular job execution on a particular device.
queuedJobs	list member: JobExecutionSummary java class: java.util.List	A list of JobExecutionSummary objects with status QUEUED.
JobExecutionSummary	JobExecutionSummary	
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.
queuedAt	long	The time, in milliseconds since the epoch, when the job execution was enqueued.

Name	Type	Description
startedAt	long java class: java.lang.Long	The time, in milliseconds since the epoch, when the job execution started.
lastUpdatedAt	long	The time, in milliseconds since the epoch, when the job execution was last updated.
versionNumber	long	The version of the job execution. Job execution versions are incremented each time AWS IoT Jobs receives an update from a device.
executionNumber	long java class: java.lang.Long	A number that identifies a particular job execution on a particular device.

GetPolicy

Gets information about the specified policy with the policy document of the default version.

Request syntax:

```
GET /policies/policyName
```

URI Request Parameters:

Name	Type	Req?	Description
policyName	PolicyName	yes	The name of the policy.

Response syntax:

```
Content-type: application/json

{
  "policyName": "string",
  "policyArn": "string",
  "policyDocument": "string",
  "defaultVersionId": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
policyName	PolicyName	no	The policy name.
policyArn	PolicyArn	no	The policy ARN.

Name	Type	Req?	Description
policyDocument	PolicyDocument	no	The JSON document that describes the policy.
defaultVersionId	PolicyVersionId	no	The default policy version ID.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  get-policy \
    --policy-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
```

```
  "policyName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The name of the policy.

Output:

```
{
  "policyName": "string",
  "policyArn": "string",
  "policyDocument": "string",
  "defaultVersionId": "string"
}
```

cli output fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.
policyArn	string	The policy ARN.
policyDocument	string	The JSON document that describes the policy.
defaultVersionId	string pattern: [0-9]+	The default policy version ID.

GetPolicyVersion

Gets information about the specified policy version.

Request syntax:

```
GET /policies/policyName/version/policyVersionId
```

URI Request Parameters:

Name	Type	Req?	Description
policyName	PolicyName	yes	The name of the policy.

Name	Type	Req?	Description
policyVersionId	PolicyVersionId	yes	The policy version ID.

Response syntax:

```
Content-type: application/json

{
  "policyArn": "string",
  "policyName": "string",
  "policyDocument": "string",
  "policyVersionId": "string",
  "isDefaultVersion": "boolean"
}
```

Response Body Parameters:

Name	Type	Req?	Description
policyArn	PolicyArn	no	The policy ARN.
policyName	PolicyName	no	The policy name.
policyDocument	PolicyDocument	no	The JSON document that describes the policy.
policyVersionId	PolicyVersionId	no	The policy version ID.
isDefaultVersion	IsDefaultVersion	no	Specifies whether the policy version is the default.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  get-policy-version \
    --policy-name <value> \
    --policy-version-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "policyName": "string",
  "policyVersionId": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The name of the policy.
policyVersionId	string pattern: [0-9]+	The policy version ID.

Output:

```
{
  "policyArn": "string",
  "policyName": "string",
  "policyDocument": "string",
  "policyVersionId": "string",
  "isDefaultVersion": "boolean"
}
```

cli output fields:

Name	Type	Description
policyArn	string	The policy ARN.

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.
policyDocument	string	The JSON document that describes the policy.
policyVersionId	string pattern: [0-9]+	The policy version ID.
isDefaultVersion	boolean	Specifies whether the policy version is the default.

GetRegistrationCode

Gets a registration code used to register a CA certificate with AWS IoT.

Request syntax:

```
GET /registrationcode
```

Response syntax:

```
Content-type: application/json

{
  "registrationCode": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
registrationCode	RegistrationCode	no	The CA certificate registration code.

Errors:

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

CLI

Synopsis:

```
aws iot  get-registration-code  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
}
```

Output:

```
{
  "registrationCode": "string"
}
```

cli output fields:

Name	Type	Description
registrationCode	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The CA certificate registration code.

GetThingShadow

Gets the thing shadow for the specified thing.

For more information, see GetThingShadow in the AWS IoT Developer Guide.

Request syntax:

```
GET /things/thingName/shadow
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	yes	The name of the thing.

Response syntax:

```
Content-type: application/json

{
  "payload": "blob"
}
```

Response Body Parameters:

Name	Type	Req?	Description
payload	JsonDocument	no	The state information, in JSON format.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`MethodNotAllowedException`

The specified combination of HTTP verb and URI is not supported.

HTTP response code: 405

`UnsupportedDocumentEncodingException`

The encoding is not supported.

HTTP response code: 415

CLI

Synopsis:

```
aws iot  get-thing-shadow \
    --thing-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing.

Output:

```
{
  "payload": "blob"
}
```

cli output fields:

Name	Type	Description
payload	blob	The state information, in JSON format.

GetTopicRule

Gets information about the rule.

Request syntax:

```
GET /rules/ruleName
```

URI Request Parameters:

Name	Type	Req?	Description
ruleName	RuleName	yes	The name of the rule.

Response syntax:

```
Content-type: application/json

{
  "ruleArn": "string",
  "rule": {
    "ruleName": "string",
    "sql": "string",
    "description": "string",
    "createdAt": "timestamp",
    "actions": [
      {
        "dynamoDB": {
          "tableName": "string",
          "roleArn": "string",
          "operation": "string",
          "hashKeyField": "string",
          "hashKeyValue": "string",
          "hashKeyType": "string",
          "rangeKeyField": "string",
          "rangeKeyValue": "string",
          "rangeKeyType": "string",
          "payloadField": "string"
        },
        "dynamoDBv2": {
          "roleArn": "string",
          "putItem": {
            "tableName": "string"
          }
        },
        "lambda": {
          "functionArn": "string"
        },
        "sns": {
          "targetArn": "string",
          "roleArn": "string",
          "messageFormat": "string"
        },
        "sqs": {
          "roleArn": "string",
          "queueUrl": "string",
          "useBase64": "boolean"
        },
        "kinesis": {
          "roleArn": "string",
          "streamName": "string",
          "partitionKey": "string"
        },
        "republish": {
          "roleArn": "string",
```

```
        "topic": "string"
      },
      "s3": {
        "roleArn": "string",
        "bucketName": "string",
        "key": "string",
        "cannedAcl": "string"
      },
      "firehose": {
        "roleArn": "string",
        "deliveryStreamName": "string",
        "separator": "string"
      },
      "cloudwatchMetric": {
        "roleArn": "string",
        "metricNamespace": "string",
        "metricName": "string",
        "metricValue": "string",
        "metricUnit": "string",
        "metricTimestamp": "string"
      },
      "cloudwatchAlarm": {
        "roleArn": "string",
        "alarmName": "string",
        "stateReason": "string",
        "stateValue": "string"
      },
      "elasticsearch": {
        "roleArn": "string",
        "endpoint": "string",
        "index": "string",
        "type": "string",
        "id": "string"
      },
      "salesforce": {
        "token": "string",
        "url": "string"
      }
    }
  ],
  "ruleDisabled": "boolean",
  "awsIotSqlVersion": "string",
  "errorAction": {
    "dynamoDB": {
      "tableName": "string",
      "roleArn": "string",
      "operation": "string",
      "hashKeyField": "string",
      "hashKeyValue": "string",
      "hashKeyType": "string",
      "rangeKeyField": "string",
      "rangeKeyValue": "string",
      "rangeKeyType": "string",
      "payloadField": "string"
    },
    "dynamoDBv2": {
      "roleArn": "string",
      "putItem": {
        "tableName": "string"
      }
    },
    "lambda": {
      "functionArn": "string"
    },
    "sns": {
      "targetArn": "string",
```

```
        "roleArn": "string",
        "messageFormat": "string"
      },
      "sqs": {
        "roleArn": "string",
        "queueUrl": "string",
        "useBase64": "boolean"
      },
      "kinesis": {
        "roleArn": "string",
        "streamName": "string",
        "partitionKey": "string"
      },
      "republish": {
        "roleArn": "string",
        "topic": "string"
      },
      "s3": {
        "roleArn": "string",
        "bucketName": "string",
        "key": "string",
        "cannedAcl": "string"
      },
      "firehose": {
        "roleArn": "string",
        "deliveryStreamName": "string",
        "separator": "string"
      },
      "cloudwatchMetric": {
        "roleArn": "string",
        "metricNamespace": "string",
        "metricName": "string",
        "metricValue": "string",
        "metricUnit": "string",
        "metricTimestamp": "string"
      },
      "cloudwatchAlarm": {
        "roleArn": "string",
        "alarmName": "string",
        "stateReason": "string",
        "stateValue": "string"
      },
      "elasticsearch": {
        "roleArn": "string",
        "endpoint": "string",
        "index": "string",
        "type": "string",
        "id": "string"
      },
      "salesforce": {
        "token": "string",
        "url": "string"
      }
    }
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
ruleArn	RuleArn	no	The rule ARN.
rule	TopicRule	no	The rule.

Errors:

`InternalException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

CLI

Synopsis:

```
aws iot  get-topic-rule \
    --rule-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "ruleName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
ruleName	string length max:128 min:1 pattern: ^[a-zA-Z0-9_]+$	The name of the rule.

Output:

```
{
  "ruleArn": "string",
  "rule": {
    "ruleName": "string",
    "sql": "string",
```

```
"description": "string",
"createdAt": "timestamp",
"actions": [
  {
    "dynamoDB": {
      "tableName": "string",
      "roleArn": "string",
      "operation": "string",
      "hashKeyField": "string",
      "hashKeyValue": "string",
      "hashKeyType": "string",
      "rangeKeyField": "string",
      "rangeKeyValue": "string",
      "rangeKeyType": "string",
      "payloadField": "string"
    },
    "dynamoDBv2": {
      "roleArn": "string",
      "putItem": {
        "tableName": "string"
      }
    },
    "lambda": {
      "functionArn": "string"
    },
    "sns": {
      "targetArn": "string",
      "roleArn": "string",
      "messageFormat": "string"
    },
    "sqs": {
      "roleArn": "string",
      "queueUrl": "string",
      "useBase64": "boolean"
    },
    "kinesis": {
      "roleArn": "string",
      "streamName": "string",
      "partitionKey": "string"
    },
    "republish": {
      "roleArn": "string",
      "topic": "string"
    },
    "s3": {
      "roleArn": "string",
      "bucketName": "string",
      "key": "string",
      "cannedAcl": "string"
    },
    "firehose": {
      "roleArn": "string",
      "deliveryStreamName": "string",
      "separator": "string"
    },
    "cloudwatchMetric": {
      "roleArn": "string",
      "metricNamespace": "string",
      "metricName": "string",
      "metricValue": "string",
      "metricUnit": "string",
      "metricTimestamp": "string"
    },
    "cloudwatchAlarm": {
      "roleArn": "string",
      "alarmName": "string",
```

```
        "stateReason": "string",
        "stateValue": "string"
      },
      "elasticsearch": {
        "roleArn": "string",
        "endpoint": "string",
        "index": "string",
        "type": "string",
        "id": "string"
      },
      "salesforce": {
        "token": "string",
        "url": "string"
      }
    }
  ],
  "ruleDisabled": "boolean",
  "awsIotSqlVersion": "string",
  "errorAction": {
    "dynamoDB": {
      "tableName": "string",
      "roleArn": "string",
      "operation": "string",
      "hashKeyField": "string",
      "hashKeyValue": "string",
      "hashKeyType": "string",
      "rangeKeyField": "string",
      "rangeKeyValue": "string",
      "rangeKeyType": "string",
      "payloadField": "string"
    },
    "dynamoDBv2": {
      "roleArn": "string",
      "putItem": {
        "tableName": "string"
      }
    },
    "lambda": {
      "functionArn": "string"
    },
    "sns": {
      "targetArn": "string",
      "roleArn": "string",
      "messageFormat": "string"
    },
    "sqs": {
      "roleArn": "string",
      "queueUrl": "string",
      "useBase64": "boolean"
    },
    "kinesis": {
      "roleArn": "string",
      "streamName": "string",
      "partitionKey": "string"
    },
    "republish": {
      "roleArn": "string",
      "topic": "string"
    },
    "s3": {
      "roleArn": "string",
      "bucketName": "string",
      "key": "string",
      "cannedAcl": "string"
    },
    "firehose": {
```

```
        "roleArn": "string",
        "deliveryStreamName": "string",
        "separator": "string"
      },
      "cloudwatchMetric": {
        "roleArn": "string",
        "metricNamespace": "string",
        "metricName": "string",
        "metricValue": "string",
        "metricUnit": "string",
        "metricTimestamp": "string"
      },
      "cloudwatchAlarm": {
        "roleArn": "string",
        "alarmName": "string",
        "stateReason": "string",
        "stateValue": "string"
      },
      "elasticsearch": {
        "roleArn": "string",
        "endpoint": "string",
        "index": "string",
        "type": "string",
        "id": "string"
      },
      "salesforce": {
        "token": "string",
        "url": "string"
      }
    }
  }
}
```

cli output fields:

Name	Type	Description
ruleArn	string	The rule ARN.
rule	TopicRule	The rule.
ruleName	string length max:128 min:1 pattern: ^[a-zA-Z0-9_]+$	The name of the rule.
sql	string	The SQL statement used to query the topic. When using a SQL query with multiple lines, be sure to escape the newline characters.
description	string	The description of the rule.
createdAt	timestamp	The date and time the rule was created.
actions	list member: Action	The actions associated with the rule.
Action	Action	

Name	Type	Description
dynamoDB	DynamoDBAction	Write to a DynamoDB table.
tableName	string	The name of the DynamoDB table.
roleArn	string	The ARN of the IAM role that grants access to the DynamoDB table.
operation	string	The type of operation to be performed. This follows the substitution template, so it can be `$ operation`, but the substitution must result in one of the following: `INSERT`, `UPDATE`, or `DELETE`.
hashKeyField	string	The hash key name.
hashKeyValue	string	The hash key value.
hashKeyType	string enum: STRING | NUMBER java class: iot.goldeneye.service.DynamoKeyType	The hash key type. Valid values are "STRING" or "NUMBER"
rangeKeyField	string	The range key name.
rangeKeyValue	string	The range key value.
rangeKeyType	string enum: STRING | NUMBER java class: iot.goldeneye.service.DynamoKeyType	The range key type. Valid values are "STRING" or "NUMBER"
payloadField	string	The action payload. This name can be customized.
dynamoDBv2	DynamoDBv2Action	Write to a DynamoDB table. This is a new version of the DynamoDB action. It allows you to write each attribute in an MQTT message payload into a separate DynamoDB column.
roleArn	string	The ARN of the IAM role that grants access to the DynamoDB table.

Name	Type	Description
putItem	PutItemInput	Specifies the DynamoDB table to which the message data will be written. For example: `{ "dynamoDBv2": { "roleArn": "aws:iam:12341251:my-role" "putItem": { "tableName": "my-table" } } }` Each attribute in the message payload will be written to a separate column in the DynamoDB database.
tableName	string	The table where the message data will be written
lambda	LambdaAction	Invoke a Lambda function.
functionArn	string	The ARN of the Lambda function.
sns	SnsAction	Publish to an Amazon SNS topic.
targetArn	string	The ARN of the SNS topic.
roleArn	string	The ARN of the IAM role that grants access.
messageFormat	string enum: RAW | JSON java class: iot.goldeneye.service.MessageFormat	The message format of the message to publish. Optional. Accepted values are "JSON" and "RAW". The default value of the attribute is "RAW". SNS uses this setting to determine if the payload should be parsed and relevant platform-specific bits of the payload should be extracted. To read more about SNS message formats, see http://docs.aws.amazon.com/sns/latest/dg/json-formats.html refer to their official documentation.
sqs	SqsAction	Publish to an Amazon SQS queue.
roleArn	string	The ARN of the IAM role that grants access.
queueUrl	string	The URL of the Amazon SQS queue.

Name	Type	Description
useBase64	boolean java class: java.lang.Boolean	Specifies whether to use Base64 encoding.
kinesis	KinesisAction	Write data to an Amazon Kinesis stream.
roleArn	string	The ARN of the IAM role that grants access to the Amazon Kinesis stream.
streamName	string	The name of the Amazon Kinesis stream.
partitionKey	string	The partition key.
republish	RepublishAction	Publish to another MQTT topic.
roleArn	string	The ARN of the IAM role that grants access.
topic	string	The name of the MQTT topic.
s3	S3Action	Write to an Amazon S3 bucket.
roleArn	string	The ARN of the IAM role that grants access.
bucketName	string	The Amazon S3 bucket.
key	string	The object key.
cannedAcl	string enum: private | public-read | public-read-write | aws-exec-read | authenticated-read | bucket-owner-read | bucket-owner-full-control | log-delivery-write java class: iot.goldeneye.service.CannedAccessControlList	The Amazon S3 canned ACL that controls access to the object identified by the object key. For more information, see S3 canned ACLs.
firehose	FirehoseAction	Write to an Amazon Kinesis Firehose stream.
roleArn	string	The IAM role that grants access to the Amazon Kinesis Firehose stream.
deliveryStreamName	string	The delivery stream name.
separator	string pattern: ([ ])|( )|(,)	A character separator that will be used to separate records written to the Firehose stream. Valid values are: '\n' (newline), '\t' (tab), '\r\n' (Windows newline), ',' (comma).

Name	Type	Description
cloudwatchMetric	CloudwatchMetricAction	Capture a CloudWatch metric.
roleArn	string	The IAM role that allows access to the CloudWatch metric.
metricNamespace	string	The CloudWatch metric namespace name.
metricName	string	The CloudWatch metric name.
metricValue	string	The CloudWatch metric value.
metricUnit	string	The metric unit supported by CloudWatch.
metricTimestamp	string	An optional Unix timestamp.
cloudwatchAlarm	CloudwatchAlarmAction	Change the state of a CloudWatch alarm.
roleArn	string	The IAM role that allows access to the CloudWatch alarm.
alarmName	string	The CloudWatch alarm name.
stateReason	string	The reason for the alarm change.
stateValue	string	The value of the alarm state. Acceptable values are: OK, ALARM, INSUFFICIENT_DATA.
elasticsearch	ElasticsearchAction	Write data to an Amazon Elasticsearch Service domain.
roleArn	string	The IAM role ARN that has access to Elasticsearch.
endpoint	string pattern: https?://.*	The endpoint of your Elasticsearch domain.
index	string	The Elasticsearch index where you want to store your data.
type	string	The type of document you are storing.
id	string	The unique identifier for the document you are storing.
salesforce	SalesforceAction	Send a message to a Salesforce IoT Cloud Input Stream.

Name	Type	Description
token	string length min:40	The token used to authenticate access to the Salesforce IoT Cloud Input Stream. The token is available from the Salesforce IoT Cloud platform after creation of the Input Stream.
url	string length max:2000 pattern: https://ingestion-[a-zA-Z0-9]{1,12}.[a-zA-Z0-9]+.((sfdc-matrix.net)|(sfdcnow.com))/streams/w *1,20*/w *1,20*/event	The URL exposed by the Salesforce IoT Cloud Input Stream. The URL is available from the Salesforce IoT Cloud platform after creation of the Input Stream.
ruleDisabled	boolean java class: java.lang.Boolean	Specifies whether the rule is disabled.
awsIotSqlVersion	string	The version of the SQL rules engine to use when evaluating the rule.
errorAction	Action	The action to perform when an error occurs.
dynamoDB	DynamoDBAction	Write to a DynamoDB table.
tableName	string	The name of the DynamoDB table.
roleArn	string	The ARN of the IAM role that grants access to the DynamoDB table.
operation	string	The type of operation to be performed. This follows the substitution template, so it can be `$ operation`, but the substitution must result in one of the following: `INSERT`, `UPDATE`, or `DELETE`.
hashKeyField	string	The hash key name.
hashKeyValue	string	The hash key value.
hashKeyType	string enum: STRING | NUMBER java class: iot.goldeneye.service.DynamoKeyType	The hash key type. Valid values are "STRING" or "NUMBER"
rangeKeyField	string	The range key name.
rangeKeyValue	string	The range key value.

Name	Type	Description
rangeKeyType	string enum: STRING | NUMBER java class: iot.goldeneye.service.DynamoKeyType	The range key type. Valid values are "STRING" or "NUMBER"
payloadField	string	The action payload. This name can be customized.
dynamoDBv2	DynamoDBv2Action	Write to a DynamoDB table. This is a new version of the DynamoDB action. It allows you to write each attribute in an MQTT message payload into a separate DynamoDB column.
roleArn	string	The ARN of the IAM role that grants access to the DynamoDB table.
putItem	PutItemInput	Specifies the DynamoDB table to which the message data will be written. For example: `{ "dynamoDBv2": { "roleArn": "aws:iam:12341251:my-role" "putItem": { "tableName": "my-table" } } }` Each attribute in the message payload will be written to a separate column in the DynamoDB database.
tableName	string	The table where the message data will be written
lambda	LambdaAction	Invoke a Lambda function.
functionArn	string	The ARN of the Lambda function.
sns	SnsAction	Publish to an Amazon SNS topic.
targetArn	string	The ARN of the SNS topic.
roleArn	string	The ARN of the IAM role that grants access.

Name	Type	Description
messageFormat	string enum: RAW | JSON java class: iot.goldeneye.service.MessageFormat	The message format of the message to publish. Optional. Accepted values are "JSON" and "RAW". The default value of the attribute is "RAW". SNS uses this setting to determine if the payload should be parsed and relevant platform-specific bits of the payload should be extracted. To read more about SNS message formats, see http://docs.aws.amazon.com/sns/latest/dg/json-formats.html refer to their official documentation.
sqs	SqsAction	Publish to an Amazon SQS queue.
roleArn	string	The ARN of the IAM role that grants access.
queueUrl	string	The URL of the Amazon SQS queue.
useBase64	boolean java class: java.lang.Boolean	Specifies whether to use Base64 encoding.
kinesis	KinesisAction	Write data to an Amazon Kinesis stream.
roleArn	string	The ARN of the IAM role that grants access to the Amazon Kinesis stream.
streamName	string	The name of the Amazon Kinesis stream.
partitionKey	string	The partition key.
republish	RepublishAction	Publish to another MQTT topic.
roleArn	string	The ARN of the IAM role that grants access.
topic	string	The name of the MQTT topic.
s3	S3Action	Write to an Amazon S3 bucket.
roleArn	string	The ARN of the IAM role that grants access.
bucketName	string	The Amazon S3 bucket.
key	string	The object key.

Name	Type	Description
cannedAcl	string enum: private | public-read | public-read-write | aws-exec-read | authenticated-read | bucket-owner-read | bucket-owner-full-control | log-delivery-write java class: iot.goldeneye.service.CannedAccessControlList	The Amazon S3 canned ACL that controls access to the object identified by the object key. For more information, see S3 canned ACLs.
firehose	FirehoseAction	Write to an Amazon Kinesis Firehose stream.
roleArn	string	The IAM role that grants access to the Amazon Kinesis Firehose stream.
deliveryStreamName	string	The delivery stream name.
separator	string pattern: ([ ])|( )|(,)	A character separator that will be used to separate records written to the Firehose stream. Valid values are: '\n' (newline), '\t' (tab), '\r\n' (Windows newline), ',' (comma).
cloudwatchMetric	CloudwatchMetricAction	Capture a CloudWatch metric.
roleArn	string	The IAM role that allows access to the CloudWatch metric.
metricNamespace	string	The CloudWatch metric namespace name.
metricName	string	The CloudWatch metric name.
metricValue	string	The CloudWatch metric value.
metricUnit	string	The metric unit supported by CloudWatch.
metricTimestamp	string	An optional Unix timestamp.
cloudwatchAlarm	CloudwatchAlarmAction	Change the state of a CloudWatch alarm.
roleArn	string	The IAM role that allows access to the CloudWatch alarm.
alarmName	string	The CloudWatch alarm name.
stateReason	string	The reason for the alarm change.

Name	Type	Description
stateValue	string	The value of the alarm state. Acceptable values are: OK, ALARM, INSUFFICIENT_DATA.
elasticsearch	ElasticsearchAction	Write data to an Amazon Elasticsearch Service domain.
roleArn	string	The IAM role ARN that has access to Elasticsearch.
endpoint	string pattern: https?://.*	The endpoint of your Elasticsearch domain.
index	string	The Elasticsearch index where you want to store your data.
type	string	The type of document you are storing.
id	string	The unique identifier for the document you are storing.
salesforce	SalesforceAction	Send a message to a Salesforce IoT Cloud Input Stream.
token	string length min:40	The token used to authenticate access to the Salesforce IoT Cloud Input Stream. The token is available from the Salesforce IoT Cloud platform after creation of the Input Stream.
url	string length max:2000 pattern: https://ingestion-[a-zA-Z0-9]{1,12}.[a-zA-Z0-9]+.((sfdc-matrix.net)|(sfdcnow.com))/streams/w *1,20*/w *1,20*/event	The URL exposed by the Salesforce IoT Cloud Input Stream. The URL is available from the Salesforce IoT Cloud platform after creation of the Input Stream.

GetV2LoggingOptions

Gets the fine grained logging options.

Request syntax:

```
GET /v2LoggingOptions
```

Response syntax:

```
Content-type: application/json
```

```
{
  "roleArn": "string",
  "defaultLogLevel": "string",
  "disableAllLogs": "boolean"
}
```

Response Body Parameters:

Name	Type	Req?	Description
roleArn	AwsArn	no	The IAM role ARN AWS IoT uses to write to your CloudWatch logs.
defaultLogLevel	LogLevel	no	The default log level.
disableAllLogs	DisableAllLogs	no	Disables all logs.

Errors:

`InternalException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  get-v2-logging-options  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
}
```

Output:

```
{
  "roleArn": "string",
  "defaultLogLevel": "string",
  "disableAllLogs": "boolean"
```

```
}
```

cli output fields:

Name	Type	Description
roleArn	string	The IAM role ARN AWS IoT uses to write to your CloudWatch logs.
defaultLogLevel	string enum: DEBUG | INFO | ERROR | WARN | DISABLED java class: iot.goldeneye.service.LogLevel	The default log level.
disableAllLogs	boolean	Disables all logs.

ListAttachedPolicies

Lists the policies attached to the specified thing group.

Request syntax:

```
POST /attached-policies/target?recursive=recursive&pageSize=pageSize&marker=marker
```

URI Request Parameters:

Name	Type	Req?	Description
target	PolicyTarget	yes	The group for which the policies will be listed.
recursive	Recursive	no	When true, recursively list attached policies.
marker	Marker	no	The token to retrieve the next set of results.
pageSize	PageSize	no	The maximum number of results to be returned per request.

Response syntax:

```
Content-type: application/json

{
  "policies": [
    {
      "policyName": "string",
      "policyArn": "string"
    }
  ],
```

```
  "nextMarker": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
policies	Policies	no	The policies.
nextMarker	Marker	no	The token to retrieve the next set of results, or `null` if there are no more results.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`LimitExceededException`

The number of attached entities exceeds the limit.

HTTP response code: 410

CLI

Synopsis:

```
aws iot  list-attached-policies \
    --target <value> \
    [--recursive | --no-recursive] \
    [--marker <value>] \
    [--page-size <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "target": "string",
  "recursive": "boolean",
  "marker": "string",
  "pageSize": "integer"
}
```

`cli-input-json` fields:

Name	Type	Description
target	string	The group for which the policies will be listed.
recursive	boolean	When true, recursively list attached policies.
marker	string pattern: [A-Za-z0-9+/]+={0,2}	The token to retrieve the next set of results.
pageSize	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to be returned per request.

Output:

```
{
  "policies": [
    {
      "policyName": "string",
      "policyArn": "string"
    }
  ],
  "nextMarker": "string"
}
```

cli output fields:

Name	Type	Description
policies	list member: Policy java class: java.util.List	The policies.

Name	Type	Description
Policy	Policy	
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.
policyArn	string	The policy ARN.
nextMarker	string pattern: [A-Za-z0-9+/]+={0,2}	The token to retrieve the next set of results, or `null` if there are no more results.

ListAuthorizers

Lists the authorizers registered in your account.

Request syntax:

```
GET /authorizers/?
pageSize=pageSize&marker=marker&isAscendingOrder=ascendingOrder&status=status
```

URI Request Parameters:

Name	Type	Req?	Description
pageSize	PageSize	no	The maximum number of results to return at one time.
marker	Marker	no	A marker used to get the next set of results.
ascendingOrder	AscendingOrder	no	Return the list of authorizers in ascending alphabetical order.
status	AuthorizerStatus	no	The status of the list authorizers request.

Response syntax:

```
Content-type: application/json

{
  "authorizers": [
    {
      "authorizerName": "string",
      "authorizerArn": "string"
    }
  ],
  "nextMarker": "string"
```

```
}
```

Response Body Parameters:

Name	Type	Req?	Description
authorizers	Authorizers	no	The authorizers.
nextMarker	Marker	no	A marker used to get the next set of results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-authorizers \
    [--page-size <value>] \
    [--marker <value>] \
    [--ascending-order | --no-ascending-order] \
    [--status <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
```

```
  "pageSize": "integer",
  "marker": "string",
  "ascendingOrder": "boolean",
  "status": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
pageSize	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to return at one time.
marker	string pattern: [A-Za-z0-9+/]+={0,2}	A marker used to get the next set of results.
ascendingOrder	boolean	Return the list of authorizers in ascending alphabetical order.
status	string enum: ACTIVE | INACTIVE java class: iot.identity.service.AuthorizerStatus	The status of the list authorizers request.

Output:

```
{
  "authorizers": [
    {
      "authorizerName": "string",
      "authorizerArn": "string"
    }
  ],
  "nextMarker": "string"
}
```

cli output fields:

Name	Type	Description
authorizers	list member: AuthorizerSummary java class: java.util.List	The authorizers.
AuthorizerSummary	AuthorizerSummary	
authorizerName	string length max:128 min:1 pattern: [w=,@-]+	The authorizer name.

Name	Type	Description
authorizerArn	string	The authorizer ARN.
nextMarker	string pattern: [A-Za-z0-9+/]+={0,2}	A marker used to get the next set of results.

ListCACertificates

Lists the CA certificates registered for your AWS account.

The results are paginated with a default page size of 25. You can use the returned marker to retrieve additional results.

Request syntax:

```
GET /cacertificates?pageSize=pageSize&marker=marker&isAscendingOrder=ascendingOrder
```

URI Request Parameters:

Name	Type	Req?	Description
pageSize	PageSize	no	The result page size.
marker	Marker	no	The marker for the next set of results.
ascendingOrder	AscendingOrder	no	Determines the order of the results.

Response syntax:

```
Content-type: application/json

{
  "certificates": [
    {
      "certificateArn": "string",
      "certificateId": "string",
      "status": "string",
      "creationDate": "timestamp"
    }
  ],
  "nextMarker": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
certificates	CACertificates	no	The CA certificates registered in your AWS account.

Name	Type	Req?	Description
nextMarker	Marker	no	The current position within the list of CA certificates.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-ca-certificates \
    [--page-size <value>] \
    [--marker <value>] \
    [--ascending-order | --no-ascending-order]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "pageSize": "integer",
  "marker": "string",
  "ascendingOrder": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
pageSize	integer java class: java.lang.Integer range- max:250 min:1	The result page size.
marker	string pattern: [A-Za-z0-9+/]+={0,2}	The marker for the next set of results.
ascendingOrder	boolean	Determines the order of the results.

Output:

```
{
  "certificates": [
    {
      "certificateArn": "string",
      "certificateId": "string",
      "status": "string",
      "creationDate": "timestamp"
    }
  ],
  "nextMarker": "string"
}
```

cli output fields:

Name	Type	Description
certificates	list member: CACertificate java class: java.util.List	The CA certificates registered in your AWS account.
CACertificate	CACertificate	
certificateArn	string	The ARN of the CA certificate.
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the CA certificate.
status	string enum: ACTIVE | INACTIVE java class: iot.identity.service.CACertificateStatus	The status of the CA certificate. The status value REGISTER_INACTIVE is deprecated and should not be used.
creationDate	timestamp	The date the CA certificate was created.

Name	Type	Description
nextMarker	string pattern: [A-Za-z0-9+/]+={0,2}	The current position within the list of CA certificates.

ListCertificates

Lists the certificates registered in your AWS account.

The results are paginated with a default page size of 25. You can use the returned marker to retrieve additional results.

Request syntax:

```
GET /certificates?pageSize=pageSize&marker=marker&isAscendingOrder=ascendingOrder
```

URI Request Parameters:

Name	Type	Req?	Description
pageSize	PageSize	no	The result page size.
marker	Marker	no	The marker for the next set of results.
ascendingOrder	AscendingOrder	no	Specifies the order for results. If True, the results are returned in ascending order, based on the creation date.

Response syntax:

```
Content-type: application/json

{
  "certificates": [
    {
      "certificateArn": "string",
      "certificateId": "string",
      "status": "string",
      "creationDate": "timestamp"
    }
  ],
  "nextMarker": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
certificates	Certificates	no	The descriptions of the certificates.

Name	Type	Req?	Description
nextMarker	Marker	no	The marker for the next set of results, or null if there are no additional results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-certificates \
    [--page-size <value>] \
    [--marker <value>] \
    [--ascending-order | --no-ascending-order]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "pageSize": "integer",
  "marker": "string",
```

```
  "ascendingOrder": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
pageSize	integer java class: java.lang.Integer range- max:250 min:1	The result page size.
marker	string pattern: [A-Za-z0-9+/]+={0,2}	The marker for the next set of results.
ascendingOrder	boolean	Specifies the order for results. If True, the results are returned in ascending order, based on the creation date.

Output:

```
{
  "certificates": [
    {
      "certificateArn": "string",
      "certificateId": "string",
      "status": "string",
      "creationDate": "timestamp"
    }
  ],
  "nextMarker": "string"
}
```

cli output fields:

Name	Type	Description
certificates	list member: Certificate java class: java.util.List	The descriptions of the certificates.
Certificate	Certificate	
certificateArn	string	The ARN of the certificate.
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)
status	string enum: ACTIVE | INACTIVE | REVOKED | PENDING_TRANSFER	The status of the certificate. The status value REGISTER_INACTIVE is

Name	Type	Description
	| REGISTER_INACTIVE | PENDING_ACTIVATION java class: iot.identity.service.CertificateStatus	deprecated and should not be used.
creationDate	timestamp	The date and time the certificate was created.
nextMarker	string pattern: [A-Za-z0-9+/]+={0,2}	The marker for the next set of results, or null if there are no additional results.

ListCertificatesByCA

List the device certificates signed by the specified CA certificate.

Request syntax:

```
GET /certificates-by-ca/caCertificateId?
pageSize=pageSize&marker=marker&isAscendingOrder=ascendingOrder
```

URI Request Parameters:

Name	Type	Req?	Description
caCertificateId	CertificateId	yes	The ID of the CA certificate. This operation will list all registered device certificate that were signed by this CA certificate.
pageSize	PageSize	no	The result page size.
marker	Marker	no	The marker for the next set of results.
ascendingOrder	AscendingOrder	no	Specifies the order for results. If True, the results are returned in ascending order, based on the creation date.

Response syntax:

```
Content-type: application/json

{
  "certificates": [
    {
      "certificateArn": "string",
      "certificateId": "string",
```

```
      "status": "string",
      "creationDate": "timestamp"
    }
  ],
  "nextMarker": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
certificates	Certificates	no	The device certificates signed by the specified CA certificate.
nextMarker	Marker	no	The marker for the next set of results, or null if there are no additional results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-certificates-by-ca \
    --ca-certificate-id <value> \
    [--page-size <value>] \
    [--marker <value>] \
```

```
    [--ascending-order | --no-ascending-order]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "caCertificateId": "string",
  "pageSize": "integer",
  "marker": "string",
  "ascendingOrder": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
caCertificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the CA certificate. This operation will list all registered device certificate that were signed by this CA certificate.
pageSize	integer java class: java.lang.Integer range- max:250 min:1	The result page size.
marker	string pattern: [A-Za-z0-9+/]+={0,2}	The marker for the next set of results.
ascendingOrder	boolean	Specifies the order for results. If True, the results are returned in ascending order, based on the creation date.

Output:

```
{
  "certificates": [
    {
      "certificateArn": "string",
      "certificateId": "string",
      "status": "string",
      "creationDate": "timestamp"
    }
  ],
  "nextMarker": "string"
}
```

cli output fields:

Name	Type	Description
certificates	list member: Certificate	The device certificates signed by the specified CA certificate.

Name	Type	Description
	java class: java.util.List	
Certificate	Certificate	
certificateArn	string	The ARN of the certificate.
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)
status	string enum: ACTIVE | INACTIVE | REVOKED | PENDING_TRANSFER | REGISTER_INACTIVE | PENDING_ACTIVATION java class: iot.identity.service.CertificateStatus	The status of the certificate. The status value REGISTER_INACTIVE is deprecated and should not be used.
creationDate	timestamp	The date and time the certificate was created.
nextMarker	string pattern: [A-Za-z0-9+/]+={0,2}	The marker for the next set of results, or null if there are no additional results.

ListIndices

Lists the search indices.

Request syntax:

```
GET /indices?nextToken=nextToken&maxResults=maxResults
```

URI Request Parameters:

Name	Type	Req?	Description
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	QueryMaxResults	no	The maximum number of results to return at one time.

Response syntax:

```
Content-type: application/json
```

```
{
  "indexNames": [
    "string"
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
indexNames	IndexNamesList	no	The index names.
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-indices \
    [--next-token <value>] \
    [--max-results <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "nextToken": "string",
  "maxResults": "integer"
}
```

`cli-input-json` fields:

Name	Type	Description
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	integer java class: java.lang.Integer range- max:500 min:1	The maximum number of results to return at one time.

Output:

```
{
  "indexNames": [
    "string"
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
indexNames	list member: IndexName java class: java.util.List	The index names.
IndexName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.

ListJobExecutionsForJob

Lists the job executions for a job.

Request syntax:

```
GET /jobs/jobId/things?status=status&maxResults=maxResults&nextToken=nextToken
```

URI Request Parameters:

Name	Type	Req?	Description
jobId	JobId	yes	The unique identifier you assigned to this job when it was created.
status	JobExecutionStatus	no	The status of the job.
maxResults	LaserMaxResults	no	The maximum number of results to be returned per request.
nextToken	NextToken	no	The token to retrieve the next set of results.

Response syntax:

```
Content-type: application/json

{
  "executionSummaries": [
    {
      "thingArn": "string",
      "jobExecutionSummary": {
        "status": "string",
        "queuedAt": "timestamp",
        "startedAt": "timestamp",
        "lastUpdatedAt": "timestamp",
        "executionNumber": "long"
      }
    }
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
executionSummaries	JobExecutionSummaryForJobList	no	A list of job execution summaries.
nextToken	NextToken	no	The token for the next set of results, or **null** if there are no additional results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  list-job-executions-for-job \
    --job-id <value> \
    [--status <value>] \
    [--max-results <value>] \
    [--next-token <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "jobId": "string",
  "status": "string",
  "maxResults": "integer",
  "nextToken": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.
status	string enum: QUEUED | IN_PROGRESS | SUCCEEDED | FAILED | REJECTED | REMOVED | CANCELED java class: com.amazonaws.iot.laser.common.JobExecutionStatus	The status of the job.

Name	Type	Description
maxResults	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to be returned per request.
nextToken	string	The token to retrieve the next set of results.

Output:

```
{
  "executionSummaries": [
    {
      "thingArn": "string",
      "jobExecutionSummary": {
        "status": "string",
        "queuedAt": "timestamp",
        "startedAt": "timestamp",
        "lastUpdatedAt": "timestamp",
        "executionNumber": "long"
      }
    }
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
executionSummaries	list member: JobExecutionSummaryForJob java class: java.util.List	A list of job execution summaries.
JobExecutionSummaryForJob	JobExecutionSummaryForJob	
thingArn	string	The ARN of the thing on which the job execution is running.
jobExecutionSummary	JobExecutionSummary	Contains a subset of information about a job execution.
status	string enum: QUEUED | IN_PROGRESS | SUCCEEDED | FAILED | REJECTED | REMOVED | CANCELED java class: com.amazonaws.iot.laser.common.JobExecutionStatus	The status of the job execution.
queuedAt	timestamp	The time, in milliseconds since the epoch, when the job execution was queued.

Name	Type	Description
startedAt	timestamp	The time, in milliseconds since the epoch, when the job execution started.
lastUpdatedAt	timestamp	The time, in milliseconds since the epoch, when the job execution was last updated.
executionNumber	long java class: java.lang.Long	A string (consisting of the digits "0" through "9") which identifies this particular job execution on this particular device. It can be used later in commands which return or update job execution information.
nextToken	string	The token for the next set of results, or **null** if there are no additional results.

ListJobExecutionsForThing

Lists the job executions for the specified thing.

Request syntax:

```
GET /things/thingName/jobs?status=status&maxResults=maxResults&nextToken=nextToken
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	yes	The thing name.
status	JobExecutionStatus	no	An optional filter that lets you search for jobs that have the specified status.
maxResults	LaserMaxResults	no	The maximum number of results to be returned per request.
nextToken	NextToken	no	The token to retrieve the next set of results.

Response syntax:

```
Content-type: application/json

{
  "executionSummaries": [
```

```
    {
      "jobId": "string",
      "jobExecutionSummary": {
        "status": "string",
        "queuedAt": "timestamp",
        "startedAt": "timestamp",
        "lastUpdatedAt": "timestamp",
        "executionNumber": "long"
      }
    }
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
executionSummaries	JobExecutionSummaryForThingList	no	A list of job execution summaries.
nextToken	NextToken	no	The token for the next set of results, or **null** if there are no additional results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  list-job-executions-for-thing \
    --thing-name <value> \
    [--status <value>] \
```

```
    [--max-results <value>] \
    [--next-token <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string",
  "status": "string",
  "maxResults": "integer",
  "nextToken": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The thing name.
status	string enum: QUEUED | IN_PROGRESS | SUCCEEDED | FAILED | REJECTED | REMOVED | CANCELED java class: com.amazonaws.iot.laser.common.JobExecutionStatus	An optional filter that lets you search for jobs that have the specified status.
maxResults	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to be returned per request.
nextToken	string	The token to retrieve the next set of results.

Output:

```
{
  "executionSummaries": [
    {
      "jobId": "string",
      "jobExecutionSummary": {
        "status": "string",
        "queuedAt": "timestamp",
        "startedAt": "timestamp",
        "lastUpdatedAt": "timestamp",
        "executionNumber": "long"
      }
    }
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
executionSummaries	list member: JobExecutionSummaryForThing java class: java.util.List	A list of job execution summaries.
JobExecutionSummaryForThing	JobExecutionSummaryForThing	
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.
jobExecutionSummary	JobExecutionSummary	Contains a subset of information about a job execution.
status	string enum: QUEUED | IN_PROGRESS | SUCCEEDED | FAILED | REJECTED | REMOVED | CANCELED java class: com.amazonaws.iot.laser.common.JobExecutionStatus	The status of the job execution.
queuedAt	timestamp	The time, in milliseconds since the epoch, when the job execution was queued.
startedAt	timestamp	The time, in milliseconds since the epoch, when the job execution started.
lastUpdatedAt	timestamp	The time, in milliseconds since the epoch, when the job execution was last updated.
executionNumber	long java class: java.lang.Long	A string (consisting of the digits "0" through "9") which identifies this particular job execution on this particular device. It can be used later in commands which return or update job execution information.
nextToken	string	The token for the next set of results, or **null** if there are no additional results.

ListJobs

Lists jobs.

Request syntax:

```
GET /jobs?
status=status&targetSelection=targetSelection&maxResults=maxResults&nextToken=nextToken&thingGroupName=
```

URI Request Parameters:

Name	Type	Req?	Description
status	JobStatus	no	An optional filter that lets you search for jobs that have the specified status.
targetSelection	TargetSelection	no	Specifies whether the job will continue to run (CONTINUOUS), or will be complete after all those things specified as targets have completed the job (SNAPSHOT). If continuous, the job may also be run on a thing when a change is detected in a target. For example, a job will run on a thing when the thing is added to a target group, even after the job was completed by all things originally in the group.
maxResults	LaserMaxResults	no	The maximum number of results to return per request.
nextToken	NextToken	no	The token to retrieve the next set of results.
thingGroupName	ThingGroupName	no	A filter that limits the returned jobs to those for the specified group.
thingGroupId	ThingGroupId	no	A filter that limits the returned jobs to those for the specified group.

Response syntax:

```
Content-type: application/json

{
  "jobs": [
    {
      "jobArn": "string",
```

```
      "jobId": "string",
      "thingGroupId": "string",
      "targetSelection": "string",
      "status": "string",
      "createdAt": "timestamp",
      "lastUpdatedAt": "timestamp",
      "completedAt": "timestamp"
    }
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
jobs	JobSummaryList	no	A list of jobs.
nextToken	NextToken	no	The token for the next set of results, or **null** if there are no additional results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  list-jobs \
    [--status <value>] \
    [--target-selection <value>] \
    [--max-results <value>] \
    [--next-token <value>] \
```

```
    [--thing-group-name <value>] \
    [--thing-group-id <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "status": "string",
  "targetSelection": "string",
  "maxResults": "integer",
  "nextToken": "string",
  "thingGroupName": "string",
  "thingGroupId": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
status	string enum: IN_PROGRESS | CANCELED | COMPLETED java class: ccom.amazonaws.iot.laser.common.JobStatus	An optional filter that lets you search for jobs that have the specified status.
targetSelection	string enum: CONTINUOUS | SNAPSHOT java class: com.amazonaws.iot.laser.TargetSelection	Specifies whether the job will continue to run (CONTINUOUS), or will be complete after all those things specified as targets have completed the job (SNAPSHOT). If continuous, the job may also be run on a thing when a change is detected in a target. For example, a job will run on a thing when the thing is added to a target group, even after the job was completed by all things originally in the group.
maxResults	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to return per request.
nextToken	string	The token to retrieve the next set of results.
thingGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	A filter that limits the returned jobs to those for the specified group.
thingGroupId	string length max:128 min:1	A filter that limits the returned jobs to those for the specified group.

Name	Type	Description
	pattern: [a-zA-Z0-9-]+	

Output:

```
{
  "jobs": [
    {
      "jobArn": "string",
      "jobId": "string",
      "thingGroupId": "string",
      "targetSelection": "string",
      "status": "string",
      "createdAt": "timestamp",
      "lastUpdatedAt": "timestamp",
      "completedAt": "timestamp"
    }
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
jobs	list member: JobSummary java class: java.util.List	A list of jobs.
JobSummary	JobSummary	
jobArn	string	The job ARN.
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.
thingGroupId	string length max:128 min:1 pattern: [a-zA-Z0-9-]+	The ID of the thing group.
targetSelection	string enum: CONTINUOUS | SNAPSHOT java class: com.amazonaws.iot.laser.TargetSelection	Specifies whether the job will continue to run (CONTINUOUS), or will be complete after all those things specified as targets have completed the job (SNAPSHOT). If continuous, the job may also be run on a thing when a change is detected in a target. For example, a job will run on a thing when the thing is added to a target group, even

Name	Type	Description
		after the job was completed by all things originally in the group.
status	string enum: IN_PROGRESS | CANCELED | COMPLETED java class: ccom.amazonaws.iot.laser.common.JobStatus	The job summary status.
createdAt	timestamp	The time, in milliseconds since the epoch, when the job was created.
lastUpdatedAt	timestamp	The time, in milliseconds since the epoch, when the job was last updated.
completedAt	timestamp	The time, in milliseconds since the epoch, when the job completed.
nextToken	string	The token for the next set of results, or **null** if there are no additional results.

ListOTAUpdates

Lists OTA updates.

Request syntax:

```
GET /otaUpdates?maxResults=maxResults&nextToken=nextToken&otaUpdateStatus=otaUpdateStatus
```

URI Request Parameters:

Name	Type	Req?	Description
maxResults	MaxResults	no	The maximum number of results to return at one time.
nextToken	NextToken	no	A token used to retreive the next set of results.
otaUpdateStatus	OTAUpdateStatus	no	The OTA update job status.

Response syntax:

```
Content-type: application/json
```

```
{
  "otaUpdates": [
    {
      "otaUpdateId": "string",
      "otaUpdateArn": "string",
      "creationDate": "timestamp"
    }
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
otaUpdates	OTAUpdatesSummary	no	A list of OTA update jobs.
nextToken	NextToken	no	A token to use to get the next set of results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  list-ota-updates \
    [--max-results <value>] \
    [--next-token <value>] \
    [--ota-update-status <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "maxResults": "integer",
  "nextToken": "string",
  "otaUpdateStatus": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
maxResults	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to return at one time.
nextToken	string	A token used to retreive the next set of results.
otaUpdateStatus	string enum: CREATE_PENDING | CREATE_IN_PROGRESS | CREATE_COMPLETE | CREATE_FAILED	The OTA update job status.

Output:

```
{
  "otaUpdates": [
    {
      "otaUpdateId": "string",
      "otaUpdateArn": "string",
      "creationDate": "timestamp"
    }
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
otaUpdates	list member: OTAUpdateSummary	A list of OTA update jobs.
OTAUpdateSummary	OTAUpdateSummary	

Name	Type	Description
otaUpdateId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The OTA update ID.
otaUpdateArn	string	The OTA update ARN.
creationDate	timestamp	The date when the OTA update was created.
nextToken	string	A token to use to get the next set of results.

ListOutgoingCertificates

Lists certificates that are being transferred but not yet accepted.

Request syntax:

```
GET /certificates-out-going?
pageSize=pageSize&marker=marker&isAscendingOrder=ascendingOrder
```

URI Request Parameters:

Name	Type	Req?	Description
pageSize	PageSize	no	The result page size.
marker	Marker	no	The marker for the next set of results.
ascendingOrder	AscendingOrder	no	Specifies the order for results. If True, the results are returned in ascending order, based on the creation date.

Response syntax:

```
Content-type: application/json

{
  "outgoingCertificates": [
    {
      "certificateArn": "string",
      "certificateId": "string",
      "transferredTo": "string",
      "transferDate": "timestamp",
      "transferMessage": "string",
      "creationDate": "timestamp"
    }
  ],
  "nextMarker": "string"
```

```
}
```

Response Body Parameters:

Name	Type	Req?	Description
outgoingCertificates	OutgoingCertificates	no	The certificates that are being transferred but not yet accepted.
nextMarker	Marker	no	The marker for the next set of results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-outgoing-certificates \
    [--page-size <value>] \
    [--marker <value>] \
    [--ascending-order | --no-ascending-order]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
```

```
  "pageSize": "integer",
  "marker": "string",
  "ascendingOrder": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
pageSize	integer java class: java.lang.Integer range- max:250 min:1	The result page size.
marker	string pattern: [A-Za-z0-9+/]+={0,2}	The marker for the next set of results.
ascendingOrder	boolean	Specifies the order for results. If True, the results are returned in ascending order, based on the creation date.

Output:

```
{
  "outgoingCertificates": [
    {
      "certificateArn": "string",
      "certificateId": "string",
      "transferredTo": "string",
      "transferDate": "timestamp",
      "transferMessage": "string",
      "creationDate": "timestamp"
    }
  ],
  "nextMarker": "string"
}
```

cli output fields:

Name	Type	Description
outgoingCertificates	list member: OutgoingCertificate java class: java.util.List	The certificates that are being transferred but not yet accepted.
OutgoingCertificate	OutgoingCertificate	
certificateArn	string	The certificate ARN.
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The certificate ID.

Name	Type	Description
transferredTo	string pattern: [0-9]{12}	The AWS account to which the transfer was made.
transferDate	timestamp	The date the transfer was initiated.
transferMessage	string length max:128	The transfer message.
creationDate	timestamp	The certificate creation date.
nextMarker	string pattern: [A-Za-z0-9+/]+={0,2}	The marker for the next set of results.

ListPolicies

Lists your policies.

Request syntax:

```
GET /policies?marker=marker&pageSize=pageSize&isAscendingOrder=ascendingOrder
```

URI Request Parameters:

Name	Type	Req?	Description
marker	Marker	no	The marker for the next set of results.
pageSize	PageSize	no	The result page size.
ascendingOrder	AscendingOrder	no	Specifies the order for results. If true, the results are returned in ascending creation order.

Response syntax:

```
Content-type: application/json

{
  "policies": [
    {
      "policyName": "string",
      "policyArn": "string"
    }
  ],
  "nextMarker": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
policies	Policies	no	The descriptions of the policies.
nextMarker	Marker	no	The marker for the next set of results, or null if there are no additional results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-policies \
    [--marker <value>] \
    [--page-size <value>] \
    [--ascending-order | --no-ascending-order]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "marker": "string",
  "pageSize": "integer",
  "ascendingOrder": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
marker	string pattern: [A-Za-z0-9+/]+={0,2}	The marker for the next set of results.
pageSize	integer java class: java.lang.Integer range- max:250 min:1	The result page size.
ascendingOrder	boolean	Specifies the order for results. If true, the results are returned in ascending creation order.

Output:

```
{
  "policies": [
    {
      "policyName": "string",
      "policyArn": "string"
    }
  ],
  "nextMarker": "string"
}
```

cli output fields:

Name	Type	Description
policies	list member: Policy java class: java.util.List	The descriptions of the policies.
Policy	Policy	
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.
policyArn	string	The policy ARN.
nextMarker	string pattern: [A-Za-z0-9+/]+={0,2}	The marker for the next set of results, or null if there are no additional results.

ListPolicyPrincipals

Lists the principals associated with the specified policy.

Note: This API is deprecated. Please use ListTargetsForPolicy instead.

Request syntax:

```
GET /policy-principals?marker=marker&pageSize=pageSize&isAscendingOrder=ascendingOrder
x-amzn-iot-policy: policyName
```

URI Request Parameters:

Name	Type	Req?	Description
policyName	PolicyName	yes	The policy name.
marker	Marker	no	The marker for the next set of results.
pageSize	PageSize	no	The result page size.
ascendingOrder	AscendingOrder	no	Specifies the order for results. If true, the results are returned in ascending creation order.

Response syntax:

```
Content-type: application/json

{
  "principals": [
    "string"
  ],
  "nextMarker": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
principals	Principals	no	The descriptions of the principals.
nextMarker	Marker	no	The marker for the next set of results, or null if there are no additional results.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-policy-principals \
    --policy-name <value> \
    [--marker <value>] \
    [--page-size <value>] \
    [--ascending-order | --no-ascending-order]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "policyName": "string",
  "marker": "string",
  "pageSize": "integer",
  "ascendingOrder": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.

Name	Type	Description
marker	string pattern: [A-Za-z0-9+/]+={0,2}	The marker for the next set of results.
pageSize	integer java class: java.lang.Integer range- max:250 min:1	The result page size.
ascendingOrder	boolean	Specifies the order for results. If true, the results are returned in ascending creation order.

Output:

```
{
  "principals": [
    "string"
  ],
  "nextMarker": "string"
}
```

cli output fields:

Name	Type	Description
principals	list member: PrincipalArn java class: java.util.List	The descriptions of the principals.
PrincipalArn	string	
nextMarker	string pattern: [A-Za-z0-9+/]+={0,2}	The marker for the next set of results, or null if there are no additional results.

ListPolicyVersions

Lists the versions of the specified policy and identifies the default version.

Request syntax:

```
GET /policies/policyName/version
```

URI Request Parameters:

Name	Type	Req?	Description
policyName	PolicyName	yes	The policy name.

Response syntax:

```
Content-type: application/json

{
  "policyVersions": [
    {
      "versionId": "string",
      "isDefaultVersion": "boolean",
      "createDate": "timestamp"
    }
  ]
}
```

Response Body Parameters:

Name	Type	Req?	Description
policyVersions	PolicyVersions	no	The policy versions.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-policy-versions \
    --policy-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "policyName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.

Output:

```
{
  "policyVersions": [
    {
      "versionId": "string",
      "isDefaultVersion": "boolean",
      "createDate": "timestamp"
    }
  ]
}
```

cli output fields:

Name	Type	Description
policyVersions	list member: PolicyVersion java class: java.util.List	The policy versions.
PolicyVersion	PolicyVersion	
versionId	string pattern: [0-9]+	The policy version ID.
isDefaultVersion	boolean	Specifies whether the policy version is the default.
createDate	timestamp	The date and time the policy was created.

ListPrincipalPolicies

Lists the policies attached to the specified principal. If you use an Cognito identity, the ID must be in AmazonCognito Identity format.

Note: This API is deprecated. Please use ListAttachedPolicies instead.

Request syntax:

```
GET /principal-policies?marker=marker&pageSize=pageSize&isAscendingOrder=ascendingOrder
x-amzn-iot-principal: principal
```

URI Request Parameters:

Name	Type	Req?	Description
principal	Principal	yes	The principal.
marker	Marker	no	The marker for the next set of results.
pageSize	PageSize	no	The result page size.
ascendingOrder	AscendingOrder	no	Specifies the order for results. If true, results are returned in ascending creation order.

Response syntax:

```
Content-type: application/json

{
  "policies": [
    {
      "policyName": "string",
      "policyArn": "string"
    }
  ],
  "nextMarker": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
policies	Policies	no	The policies.
nextMarker	Marker	no	The marker for the next set of results, or null if there are no additional results.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-principal-policies \
    --principal <value> \
    [--marker <value>] \
    [--page-size <value>] \
    [--ascending-order | --no-ascending-order]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "principal": "string",
  "marker": "string",
  "pageSize": "integer",
  "ascendingOrder": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
principal	string	The principal.

Name	Type	Description
marker	string pattern: [A-Za-z0-9+/]+={0,2}	The marker for the next set of results.
pageSize	integer java class: java.lang.Integer range- max:250 min:1	The result page size.
ascendingOrder	boolean	Specifies the order for results. If true, results are returned in ascending creation order.

Output:

```
{
  "policies": [
    {
      "policyName": "string",
      "policyArn": "string"
    }
  ],
  "nextMarker": "string"
}
```

cli output fields:

Name	Type	Description
policies	list member: Policy java class: java.util.List	The policies.
Policy	Policy	
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.
policyArn	string	The policy ARN.
nextMarker	string pattern: [A-Za-z0-9+/]+={0,2}	The marker for the next set of results, or null if there are no additional results.

ListPrincipalThings

Lists the things associated with the specified principal.

Request syntax:

```
GET /principals/things?maxResults=maxResults&nextToken=nextToken
x-amzn-principal: principal
```

URI Request Parameters:

Name	Type	Req?	Description
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	RegistryMaxResults	no	The maximum number of results to return in this operation.
principal	Principal	yes	The principal.

Response syntax:

```
Content-type: application/json

{
  "things": [
    "string"
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
things	ThingNameList	no	The things.
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  list-principal-things \
    [--next-token <value>] \
    [--max-results <value>] \
    --principal <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "nextToken": "string",
  "maxResults": "integer",
  "principal": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to return in this operation.
principal	string	The principal.

Output:

```
{
  "things": [
    "string"
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
things	list member: ThingName	The things.
ThingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.

ListRoleAliases

Lists the role aliases registered in your account.

Request syntax:

```
GET /role-aliases?pageSize=pageSize&marker=marker&isAscendingOrder=ascendingOrder
```

URI Request Parameters:

Name	Type	Req?	Description
pageSize	PageSize	no	The maximum number of results to return at one time.
marker	Marker	no	A marker used to get the next set of results.
ascendingOrder	AscendingOrder	no	Return the list of role aliases in ascending alphabetical order.

Response syntax:

```
Content-type: application/json

{
  "roleAliases": [
```

```
    "string"
  ],
  "nextMarker": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
roleAliases	RoleAliases	no	The role aliases.
nextMarker	Marker	no	A marker used to get the next set of results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-role-aliases \
    [--page-size <value>] \
    [--marker <value>] \
    [--ascending-order | --no-ascending-order]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "pageSize": "integer",
  "marker": "string",
  "ascendingOrder": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
pageSize	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to return at one time.
marker	string pattern: [A-Za-z0-9+/]+={0,2}	A marker used to get the next set of results.
ascendingOrder	boolean	Return the list of role aliases in ascending alphabetical order.

Output:

```
{
  "roleAliases": [
    "string"
  ],
  "nextMarker": "string"
}
```

cli output fields:

Name	Type	Description
roleAliases	list member: RoleAlias java class: java.util.List	The role aliases.
RoleAlias	string length max:128 min:1 pattern: [w=,@-]+	
nextMarker	string pattern: [A-Za-z0-9+/]+={0,2}	A marker used to get the next set of results.

ListStreams

Lists all of the streams in your AWS account.

Request syntax:

```
GET /streams?maxResults=maxResults&nextToken=nextToken&isAscendingOrder=ascendingOrder
```

URI Request Parameters:

Name	Type	Req?	Description
maxResults	MaxResults	no	The maximum number of results to return at a time.
nextToken	NextToken	no	A token used to get the next set of results.
ascendingOrder	AscendingOrder	no	Set to true to return the list of streams in ascending order.

Response syntax:

```
Content-type: application/json

{
  "streams": [
    {
      "streamId": "string",
      "streamArn": "string",
      "streamVersion": "integer",
      "description": "string"
    }
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
streams	StreamsSummary	no	A list of streams.
nextToken	NextToken	no	A token used to get the next set of results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-streams \
    [--max-results <value>] \
    [--next-token <value>] \
    [--ascending-order | --no-ascending-order]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "maxResults": "integer",
  "nextToken": "string",
  "ascendingOrder": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
maxResults	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to return at a time.
nextToken	string	A token used to get the next set of results.
ascendingOrder	boolean	Set to true to return the list of streams in ascending order.

Output:

```
{
  "streams": [
    {
```

```
      "streamId": "string",
      "streamArn": "string",
      "streamVersion": "integer",
      "description": "string"
    }
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
streams	list member: StreamSummary	A list of streams.
StreamSummary	StreamSummary	
streamId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The stream ID.
streamArn	string	The stream ARN.
streamVersion	integer java class: java.lang.Integer range- max:65535 min:0	The stream version.
description	string length max:2028 pattern: [^\\p{C}]+	A description of the stream.
nextToken	string	A token used to get the next set of results.

ListTargetsForPolicy

List targets for the specified policy.

Request syntax:

```
POST /policy-targets/policyName?marker=marker&pageSize=pageSize
```

URI Request Parameters:

Name	Type	Req?	Description
policyName	PolicyName	yes	The policy name.
marker	Marker	no	A marker used to get the next set of results.

Name	Type	Req?	Description
pageSize	PageSize	no	The maximum number of results to return at one time.

Response syntax:

```
Content-type: application/json

{
  "targets": [
    "string"
  ],
  "nextMarker": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
targets	PolicyTargets	no	The policy targets.
nextMarker	Marker	no	A marker used to get the next set of results.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`LimitExceededException`

The number of attached entities exceeds the limit.

HTTP response code: 410

CLI

Synopsis:

```
aws iot  list-targets-for-policy \
    --policy-name <value> \
    [--marker <value>] \
    [--page-size <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "policyName": "string",
  "marker": "string",
  "pageSize": "integer"
}
```

`cli-input-json` fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.
marker	string pattern: [A-Za-z0-9+/]+={0,2}	A marker used to get the next set of results.
pageSize	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to return at one time.

Output:

```
{
  "targets": [
    "string"
  ],
  "nextMarker": "string"
}
```

cli output fields:

Name	Type	Description
targets	list member: PolicyTarget java class: java.util.List	The policy targets.
PolicyTarget	string	
nextMarker	string pattern: [A-Za-z0-9+/]+={0,2}	A marker used to get the next set of results.

ListThingGroups

List the thing groups in your account.

Request syntax:

```
GET /thing-groups?
maxResults=maxResults&nextToken=nextToken&parentGroup=parentGroup&namePrefixFilter=namePrefixFilter&rec
```

URI Request Parameters:

Name	Type	Req?	Description
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	RegistryMaxResults	no	The maximum number of results to return at one time.
parentGroup	ThingGroupName	no	A filter that limits the results to those with the specified parent group.
namePrefixFilter	ThingGroupName	no	A filter that limits the results to those with the specified name prefix.
recursive	RecursiveWithoutDefault	no	If true, return child groups as well.

Response syntax:

```
Content-type: application/json
```

```
{
  "thingGroups": [
    {
      "groupName": "string",
      "groupArn": "string"
    }
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
thingGroups	ThingGroupNameAndArnList	no	The thing groups.
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  list-thing-groups \
    [--next-token <value>] \
    [--max-results <value>] \
    [--parent-group <value>] \
    [--name-prefix-filter <value>] \
    [--recursive | --no-recursive]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
```

```
  "nextToken": "string",
  "maxResults": "integer",
  "parentGroup": "string",
  "namePrefixFilter": "string",
  "recursive": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to return at one time.
parentGroup	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	A filter that limits the results to those with the specified parent group.
namePrefixFilter	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	A filter that limits the results to those with the specified name prefix.
recursive	boolean java class: java.lang.Boolean	If true, return child groups as well.

Output:

```
{
  "thingGroups": [
    {
      "groupName": "string",
      "groupArn": "string"
    }
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
thingGroups	list member: GroupNameAndArn java class: java.util.List	The thing groups.
GroupNameAndArn	GroupNameAndArn	

Name	Type	Description
groupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The group name.
groupArn	string	The group ARN.
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.

ListThingGroupsForThing

List the thing groups to which the specified thing belongs.

Request syntax:

```
GET /things/thingName/thing-groups?maxResults=maxResults&nextToken=nextToken
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	yes	The thing name.
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	RegistryMaxResults	no	The maximum number of results to return at one time.

Response syntax:

```
Content-type: application/json

{
  "thingGroups": [
    {
      "groupName": "string",
      "groupArn": "string"
    }
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
thingGroups	ThingGroupNameAndArnList		The thing groups.

Name	Type	Req?	Description
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  list-thing-groups-for-thing \
    --thing-name <value> \
    [--next-token <value>] \
    [--max-results <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string",
  "nextToken": "string",
  "maxResults": "integer"
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The thing name.

Name	Type	Description
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to return at one time.

Output:

```
{
  "thingGroups": [
    {
      "groupName": "string",
      "groupArn": "string"
    }
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
thingGroups	list member: GroupNameAndArn java class: java.util.List	The thing groups.
GroupNameAndArn	GroupNameAndArn	
groupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The group name.
groupArn	string	The group ARN.
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.

ListThingPrincipals

Lists the principals associated with the specified thing.

Request syntax:

```
GET /things/thingName/principals
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	yes	The name of the thing.

Response syntax:

```
Content-type: application/json

{
  "principals": [
    "string"
  ]
}
```

Response Body Parameters:

Name	Type	Req?	Description
principals	Principals	no	The principals associated with the thing.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  list-thing-principals \
    --thing-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing.

Output:

```
{
  "principals": [
    "string"
  ]
}
```

cli output fields:

Name	Type	Description
principals	list member: PrincipalArn java class: java.util.List	The principals associated with the thing.
PrincipalArn	string	

ListThingRegistrationTaskReports

Information about the thing registration tasks.

Request syntax:

```
GET /thing-registration-tasks/taskId/reports?
reportType=reportType&maxResults=maxResults&nextToken=nextToken
```

URI Request Parameters:

Name	Type	Req?	Description
taskId	TaskId	yes	The id of the task.
reportType	ReportType	yes	The type of task report.
nextToken	NextToken	no	The token to retrieve the next set of results.
maxResults	RegistryMaxResults	no	The maximum number of results to return per request.

Response syntax:

```
Content-type: application/json

{
  "resourceLinks": [
    "string"
  ],
  "reportType": "string",
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
resourceLinks	S3FileUrlList	no	Links to the task resources.
reportType	ReportType	no	The type of task report.
nextToken	NextToken	no	The token to retrieve the next set of results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-thing-registration-task-reports \
    --task-id <value> \
    --report-type <value> \
    [--next-token <value>] \
    [--max-results <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "taskId": "string",
  "reportType": "string",
  "nextToken": "string",
  "maxResults": "integer"
}
```

`cli-input-json` fields:

Name	Type	Description
taskId	string length max:40	The id of the task.
reportType	string enum: ERRORS | RESULTS java class: com.amazonaws.iot.common.types.enums.ReportType	The type of task report.
nextToken	string	The token to retrieve the next set of results.
maxResults	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to return per request.

Output:

```
{
  "resourceLinks": [
    "string"
```

```
  ],
  "reportType": "string",
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
resourceLinks	list member: S3FileUrl	Links to the task resources.
S3FileUrl	string length max:65535	
reportType	string enum: ERRORS | RESULTS java class: com.amazonaws.iot.common.types.enums.ReportType	The type of task report.
nextToken	string	The token to retrieve the next set of results.

ListThingRegistrationTasks

List bulk thing provisioning tasks.

Request syntax:

```
GET /thing-registration-tasks?maxResults=maxResults&nextToken=nextToken&status=status
```

URI Request Parameters:

Name	Type	Req?	Description
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	RegistryMaxResults	no	The maximum number of results to return at one time.
status	Status	no	The status of the bulk thing provisioning task.

Response syntax:

```
Content-type: application/json
```

```
{
  "taskIds": [
    "string"
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
taskIds	TaskIdList	no	A list of bulk thing provisioning task IDs.
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-thing-registration-tasks \
    [--next-token <value>] \
    [--max-results <value>] \
    [--status <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "nextToken": "string",
  "maxResults": "integer",
  "status": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to return at one time.
status	string enum: InProgress | Completed | Failed | Cancelled | Cancelling java class: com.amazonaws.iot.common.types.enums.Status	The status of the bulk thing provisioning task.

Output:

```
{
  "taskIds": [
    "string"
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
taskIds	list member: TaskId	A list of bulk thing provisioning task IDs.
TaskId	string length max:40	
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.

ListThingTypes

Lists the existing thing types.

Request syntax:

```
GET /thing-types?maxResults=maxResults&nextToken=nextToken&thingTypeName=thingTypeName
```

URI Request Parameters:

Name	Type	Req?	Description
nextToken	NextToken	no	The token for the next set of results, or **null** if there are no additional results.
maxResults	RegistryMaxResults	no	The maximum number of results to return in this operation.
thingTypeName	ThingTypeName	no	The name of the thing type.

Response syntax:

```
Content-type: application/json

{
  "thingTypes": [
    {
      "thingTypeName": "string",
      "thingTypeArn": "string",
      "thingTypeProperties": {
        "thingTypeDescription": "string",
        "searchableAttributes": [
          "string"
        ]
      },
      "thingTypeMetadata": {
        "deprecated": "boolean",
        "deprecationDate": "timestamp",
        "creationDate": "timestamp"
      }
    }
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
thingTypes	ThingTypeList	no	The thing types.
nextToken	NextToken	no	The token for the next set of results, or **null** if there are no additional results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-thing-types \
    [--next-token <value>] \
    [--max-results <value>] \
    [--thing-type-name <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "nextToken": "string",
  "maxResults": "integer",
  "thingTypeName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
nextToken	string	The token for the next set of results, or **null** if there are no additional results.
maxResults	integer java class: java.lang.Integer	The maximum number of results to return in this operation.

Name	Type	Description
	range- max:250 min:1	
thingTypeName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing type.

Output:

```
{
  "thingTypes": [
    {
      "thingTypeName": "string",
      "thingTypeArn": "string",
      "thingTypeProperties": {
        "thingTypeDescription": "string",
        "searchableAttributes": [
          "string"
        ]
      },
      "thingTypeMetadata": {
        "deprecated": "boolean",
        "deprecationDate": "timestamp",
        "creationDate": "timestamp"
      }
    }
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
thingTypes	list member: ThingTypeDefinition java class: java.util.List	The thing types.
ThingTypeDefinition	ThingTypeDefinition	
thingTypeName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing type.
thingTypeArn	string	The thing type ARN.
thingTypeProperties	ThingTypeProperties	The ThingTypeProperties for the thing type.
thingTypeDescription	string length max:2028 pattern: [\\p{Graph}]*	The description of the thing type.

Name	Type	Description
searchableAttributes	list member: AttributeName java class: java.util.List	A list of searchable thing attribute names.
AttributeName	string length max:128 pattern: [a-zA-Z0-9_.,@/:#-]+	
thingTypeMetadata	ThingTypeMetadata	The ThingTypeMetadata contains additional information about the thing type including: creation date and time, a value indicating whether the thing type is deprecated, and a date and time when it was deprecated.
deprecated	boolean	Whether the thing type is deprecated. If **true**, no new things could be associated with this type.
deprecationDate	timestamp	The date and time when the thing type was deprecated.
creationDate	timestamp	The date and time when the thing type was created.
nextToken	string	The token for the next set of results, or **null** if there are no additional results.

ListThings

Lists your things. Use the **attributeName** and **attributeValue** parameters to filter your things. For example, calling `ListThings` with attributeName=Color and attributeValue=Red retrieves all things in the registry that contain an attribute **Color** with the value **Red**.

Request syntax:

```
GET /things?
maxResults=maxResults&nextToken=nextToken&attributeName=attributeName&attributeValue=attributeValue&thi
```

URI Request Parameters:

Name	Type	Req?	Description
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.

Name	Type	Req?	Description
maxResults	RegistryMaxResults	no	The maximum number of results to return in this operation.
attributeName	AttributeName	no	The attribute name used to search for things.
attributeValue	AttributeValue	no	The attribute value used to search for things.
thingTypeName	ThingTypeName	no	The name of the thing type used to search for things.

Response syntax:

```
Content-type: application/json

{
  "things": [
    {
      "thingName": "string",
      "thingTypeName": "string",
      "thingArn": "string",
      "attributes": {
        "string": "string"
      },
      "version": "long"
    }
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
things	ThingAttributeList	no	The things.
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  list-things \
    [--next-token <value>] \
    [--max-results <value>] \
    [--attribute-name <value>] \
    [--attribute-value <value>] \
    [--thing-type-name <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "nextToken": "string",
  "maxResults": "integer",
  "attributeName": "string",
  "attributeValue": "string",
  "thingTypeName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to return in this operation.

Name	Type	Description
attributeName	string length max:128 pattern: [a-zA-Z0-9_.,@/:#-]+	The attribute name used to search for things.
attributeValue	string length max:800 pattern: [a-zA-Z0-9_.,@/:#-]*	The attribute value used to search for things.
thingTypeName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing type used to search for things.

Output:

```
{
  "things": [
    {
      "thingName": "string",
      "thingTypeName": "string",
      "thingArn": "string",
      "attributes": {
        "string": "string"
      },
      "version": "long"
    }
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
things	list member: ThingAttribute java class: java.util.List	The things.
ThingAttribute	ThingAttribute	
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing.
thingTypeName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing type, if the thing has been associated with a type.

Name	Type	Description
thingArn	string	The thing ARN.
attributes	map key: AttributeName value: AttributeValue	A list of thing attributes which are name-value pairs.
AttributeName	string length max:128 pattern: [a-zA-Z0-9_.,@/:#-]+	
AttributeValue	string length max:800 pattern: [a-zA-Z0-9_.,@/:#-]*	
version	long	The version of the thing record in the registry.
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.

ListThingsInThingGroup

Lists the things in the specified group.

Request syntax:

```
GET /thing-groups/thingGroupName/things?
recursive=recursive&maxResults=maxResults&nextToken=nextToken
```

URI Request Parameters:

Name	Type	Req?	Description
thingGroupName	ThingGroupName	yes	The thing group name.
recursive	Recursive	no	When true, list things in this thing group and in all child groups as well.
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	RegistryMaxResults	no	The maximum number of results to return at one time.

Response syntax:

```
Content-type: application/json

{
  "things": [
    "string"
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
things	ThingNameList	no	The things in the specified thing group.
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  list-things-in-thing-group \
    --thing-group-name <value> \
    [--recursive | --no-recursive] \
    [--next-token <value>] \
    [--max-results <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingGroupName": "string",
  "recursive": "boolean",
  "nextToken": "string",
  "maxResults": "integer"
}
```

`cli-input-json` fields:

Name	Type	Description
thingGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The thing group name.
recursive	boolean	When true, list things in this thing group and in all child groups as well.
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to return at one time.

Output:

```
{
  "things": [
    "string"
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
things	list member: ThingName	The things in the specified thing group.
ThingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.

ListTopicRules

Lists the rules for the specific topic.

Request syntax:

```
GET /rules?topic=topic&maxResults=maxResults&nextToken=nextToken&ruleDisabled=ruleDisabled
```

URI Request Parameters:

Name	Type	Req?	Description
topic	Topic	no	The topic.
maxResults	GEMaxResults	no	The maximum number of results to return.
nextToken	NextToken	no	A token used to retrieve the next value.
ruleDisabled	IsDisabled	no	Specifies whether the rule is disabled.

Response syntax:

```
Content-type: application/json

{
  "rules": [
    {
      "ruleArn": "string",
      "ruleName": "string",
      "topicPattern": "string",
      "createdAt": "timestamp",
      "ruleDisabled": "boolean"
    }
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
rules	TopicRuleList	no	The rules.
nextToken	NextToken	no	A token used to retrieve the next value.

Errors:

`InternalException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  list-topic-rules \
    [--topic <value>] \
    [--max-results <value>] \
    [--next-token <value>] \
    [--rule-disabled | --no-rule-disabled]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "topic": "string",
  "maxResults": "integer",
  "nextToken": "string",
  "ruleDisabled": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
topic	string	The topic.
maxResults	integer java class: java.lang.Integer range- max:10000 min:1	The maximum number of results to return.
nextToken	string	A token used to retrieve the next value.
ruleDisabled	boolean java class: java.lang.Boolean	Specifies whether the rule is disabled.

Output:

```
{
  "rules": [
    {
      "ruleArn": "string",
      "ruleName": "string",
      "topicPattern": "string",
      "createdAt": "timestamp",
      "ruleDisabled": "boolean"
    }
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
rules	list member: TopicRuleListItem	The rules.
TopicRuleListItem	TopicRuleListItem	
ruleArn	string	The rule ARN.
ruleName	string length max:128 min:1 pattern: ^[a-zA-Z0-9_]+$	The name of the rule.
topicPattern	string	The pattern for the topic names that apply.
createdAt	timestamp	The date and time the rule was created.
ruleDisabled	boolean java class: java.lang.Boolean	Specifies whether the rule is disabled.
nextToken	string	A token used to retrieve the next value.

ListV2LoggingLevels

Lists logging levels.

Request syntax:

```
GET /v2LoggingLevel?maxResults=maxResults&nextToken=nextToken&targetType=targetType
```

URI Request Parameters:

Name	Type	Req?	Description
targetType	LogTargetType	no	The type of resource for which you are

Name	Type	Req?	Description
			configuring logging. Must be `THING_Group`.
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	SkyfallMaxResults	no	The maximum number of results to return at one time.

Response syntax:

```
Content-type: application/json

{
  "logTargetConfigurations": [
    {
      "logTarget": {
        "targetType": "string",
        "targetName": "string"
      },
      "logLevel": "string"
    }
  ],
  "nextToken": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
logTargetConfigurations	LogTargetConfigurations	no	The logging configuration for a target.
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.

Errors:

`InternalException`

An unexpected error has occurred.

HTTP response code: 500

`NotConfiguredException`

The resource is not configured.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  list-v2-logging-levels \
    [--target-type <value>] \
    [--next-token <value>] \
    [--max-results <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "targetType": "string",
  "nextToken": "string",
  "maxResults": "integer"
}
```

`cli-input-json` fields:

Name	Type	Description
targetType	string enum: DEFAULT | THING_GROUP java class: iot.goldeneye.service.LogTargetType	The type of resource for which you are configuring logging. Must be `THING_Group`.
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	integer java class: java.lang.Integer range- max:250 min:1	The maximum number of results to return at one time.

Output:

```
{
  "logTargetConfigurations": [
```

```
    {
      "logTarget": {
        "targetType": "string",
        "targetName": "string"
      },
      "logLevel": "string"
    }
  ],
  "nextToken": "string"
}
```

cli output fields:

Name	Type	Description
logTargetConfigurations	list member: LogTargetConfiguration	The logging configuration for a target.
LogTargetConfiguration	LogTargetConfiguration	
logTarget	LogTarget	A log target
targetType	string enum: DEFAULT | THING_GROUP java class: iot.goldeneye.service.LogTargetType	The target type.
targetName	string	The target name.
logLevel	string enum: DEBUG | INFO | ERROR | WARN | DISABLED java class: iot.goldeneye.service.LogLevel	The logging level.
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.

Publish

Publishes state information.

For more information, see HTTP Protocol in the AWS IoT Developer Guide.

Request syntax:

```
POST /topics/topic?qos=qos
Content-type: application/json

{
  "payload": "blob"
}
```

URI Request Parameters:

Name	Type	Req?	Description
topic	Topic	no	The name of the MQTT topic.
qos	Qos	no	The Quality of Service (QoS) level.

Request Body Parameters:

Name	Type	Req?	Description
payload	Payload	no	The state information, in JSON format.

Errors:

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`MethodNotAllowedException`

The specified combination of HTTP verb and URI is not supported.

HTTP response code: 405

CLI

Synopsis:

```
aws iot  publish \
    [--topic <value>] \
    [--qos <value>] \
    [--payload <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
```

```
  "topic": "string",
  "qos": "integer",
  "payload": "blob"
}
```

`cli-input-json` fields:

Name	Type	Description
topic	string	The name of the MQTT topic.
qos	integer range- max:1 min:0	The Quality of Service (QoS) level.
payload	blob	The state information, in JSON format.

Output:

None

RegisterCACertificate

Registers a CA certificate with AWS IoT. This CA certificate can then be used to sign device certificates, which can be then registered with AWS IoT. You can register up to 10 CA certificates per AWS account that have the same subject field. This enables you to have up to 10 certificate authorities sign your device certificates. If you have more than one CA certificate registered, make sure you pass the CA certificate when you register your device certificates with the RegisterCertificate API.

Request syntax:

```
POST /cacertificate?setAsActive=setAsActive&allowAutoRegistration=allowAutoRegistration
Content-type: application/json

{
  "caCertificate": "string",
  "verificationCertificate": "string",
  "registrationConfig": {
    "templateBody": "string",
    "roleArn": "string"
  }
}
```

URI Request Parameters:

Name	Type	Req?	Description
setAsActive	SetAsActive	no	A boolean value that specifies if the CA certificate is set to active.
allowAutoRegistration	AllowAutoRegistration	no	Allows this CA certificate to be used for auto registration of device certificates.

Request Body Parameters:

Name	Type	Req?	Description
caCertificate	CertificatePem	yes	The CA certificate.
verificationCertificate	CertificatePem	yes	The private key verification certificate.
registrationConfig	RegistrationConfig	no	Information about the registration configuration.

Response syntax:

```
Content-type: application/json

{
  "certificateArn": "string",
  "certificateId": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
certificateArn	CertificateArn	no	The CA certificate ARN.
certificateId	CertificateId	no	The CA certificate identifier.

Errors:

`ResourceAlreadyExistsException`

The resource already exists.

HTTP response code: 409

`RegistrationCodeValidationException`

The registration code is invalid.

HTTP response code: 400

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`CertificateValidationException`

The certificate is invalid.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`LimitExceededException`

The number of attached entities exceeds the limit.

HTTP response code: 410

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  register-ca-certificate \
    --ca-certificate <value> \
    --verification-certificate <value> \
    [--set-as-active | --no-set-as-active] \
    [--allow-auto-registration | --no-allow-auto-registration] \
    [--registration-config <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "caCertificate": "string",
  "verificationCertificate": "string",
  "setAsActive": "boolean",
  "allowAutoRegistration": "boolean",
  "registrationConfig": {
    "templateBody": "string",
    "roleArn": "string"
  }
}
```

`cli-input-json` fields:

Name	Type	Description
caCertificate	string length max:65536 min:1	The CA certificate.

Name	Type	Description
verificationCertificate	string length max:65536 min:1	The private key verification certificate.
setAsActive	boolean	A boolean value that specifies if the CA certificate is set to active.
allowAutoRegistration	boolean	Allows this CA certificate to be used for auto registration of device certificates.
registrationConfig	RegistrationConfig	Information about the registration configuration.
templateBody	string	The template body.
roleArn	string length max:2048 min:20	The ARN of the role.

Output:

```
{
  "certificateArn": "string",
  "certificateId": "string"
}
```

cli output fields:

Name	Type	Description
certificateArn	string	The CA certificate ARN.
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The CA certificate identifier.

RegisterCertificate

Registers a device certificate with AWS IoT. If you have more than one CA certificate that has the same subject field, you must specify the CA certificate that was used to sign the device certificate being registered.

Request syntax:

```
POST /certificate/register?setAsActive=setAsActive
Content-type: application/json

{
  "certificatePem": "string",
  "caCertificatePem": "string",
  "status": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
setAsActive	SetAsActiveFlag	no	A boolean value that specifies if the CA certificate is set to active.

Request Body Parameters:

Name	Type	Req?	Description
certificatePem	CertificatePem	yes	The certificate data, in PEM format.
caCertificatePem	CertificatePem	no	The CA certificate used to sign the device certificate being registered.
status	CertificateStatus	no	The status of the register certificate request.

Response syntax:

```
Content-type: application/json

{
  "certificateArn": "string",
  "certificateId": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
certificateArn	CertificateArn	no	The certificate ARN.
certificateId	CertificateId	no	The certificate identifier.

Errors:

`ResourceAlreadyExistsException`

The resource already exists.

HTTP response code: 409

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`CertificateValidationException`

The certificate is invalid.

HTTP response code: 400

`CertificateStateException`

The certificate operation is not allowed.

HTTP response code: 406

`CertificateConflictException`

Unable to verify the CA certificate used to sign the device certificate you are attempting to register. This is happens when you have registered more than one CA certificate that has the same subject field and public key.

HTTP response code: 409

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  register-certificate \
    --certificate-pem <value> \
    [--ca-certificate-pem <value>] \
    [--set-as-active | --no-set-as-active] \
    [--status <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "certificatePem": "string",
  "caCertificatePem": "string",
  "status": "string"
```

```
}
```

`cli-input-json` fields:

Name	Type	Description
certificatePem	string length max:65536 min:1	The certificate data, in PEM format.
caCertificatePem	string length max:65536 min:1	The CA certificate used to sign the device certificate being registered.
status	string enum: ACTIVE | INACTIVE | REVOKED | PENDING_TRANSFER | REGISTER_INACTIVE | PENDING_ACTIVATION java class: iot.identity.service.CertificateStatus	The status of the register certificate request.

Output:

```
{
  "certificateArn": "string",
  "certificateId": "string"
}
```

cli output fields:

Name	Type	Description
certificateArn	string	The certificate ARN.
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The certificate identifier.

RegisterThing

Provisions a thing.

Request syntax:

```
POST /things
Content-type: application/json

{
  "templateBody": "string",
  "parameters": {
    "string": "string"
```

```
  }
}
```

Request Body Parameters:

Name	Type	Req?	Description
templateBody	TemplateBody	yes	The provisioning template.
parameters	Parameters	no	The parameters for provisioning a thing.

Response syntax:

```
Content-type: application/json

{
  "certificatePem": "string",
  "resourceArns": {
    "string": "string"
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
certificatePem	CertificatePem	no	
resourceArns	ResourceArns	no	ARNs for the generated resources.

Errors:

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ConflictingResourceUpdateException`

A conflicting resource update exception. This exception is thrown when two pending updates cause a conflict.

HTTP response code: 409

`ResourceRegistrationFailureException`

The resource registration failed.

HTTP response code: 400

CLI

Synopsis:

```
aws iot  register-thing \
    --template-body <value> \
    [--parameters <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "templateBody": "string",
  "parameters": {
    "string": "string"
  }
}
```

`cli-input-json` fields:

Name	Type	Description
templateBody	string	The provisioning template.
parameters	map key: Parameter value: Value	The parameters for provisioning a thing.
Parameter	string	
Value	string	

Output:

```
{
  "certificatePem": "string",
  "resourceArns": {
```

```
    "string": "string"
  }
}
```

cli output fields:

Name	Type	Description
certificatePem	string length max:65536 min:1	
resourceArns	map key: ResourceLogicalId value: ResourceArn	ARNs for the generated resources.
ResourceLogicalId	string	
ResourceArn	string	

RejectCertificateTransfer

Rejects a pending certificate transfer. After AWS IoT rejects a certificate transfer, the certificate status changes from **PENDING_TRANSFER** to **INACTIVE**.

To check for pending certificate transfers, call ListCertificates to enumerate your certificates.

This operation can only be called by the transfer destination. After it is called, the certificate will be returned to the source's account in the INACTIVE state.

Request syntax:

```
PATCH /reject-certificate-transfer/certificateId
Content-type: application/json

{
  "rejectReason": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
certificateId	CertificateId	yes	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)

Request Body Parameters:

Name	Type	Req?	Description
rejectReason	Message	no	The reason the certificate transfer was rejected.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`TransferAlreadyCompletedException`

You can't revert the certificate transfer because the transfer is already complete.

HTTP response code: 410

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  reject-certificate-transfer \
    --certificate-id <value> \
    [--reject-reason <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "certificateId": "string",
  "rejectReason": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)
rejectReason	string length max:128	The reason the certificate transfer was rejected.

Output:

None

RemoveThingFromThingGroup

Remove the specified thing from the specified group.

Request syntax:

```
PUT /thing-groups/removeThingFromThingGroup
Content-type: application/json

{
  "thingGroupName": "string",
  "thingGroupArn": "string",
  "thingName": "string",
  "thingArn": "string"
}
```

Request Body Parameters:

Name	Type	Req?	Description
thingGroupName	ThingGroupName	no	The group name.
thingGroupArn	ThingGroupArn	no	The group ARN.
thingName	ThingName	no	The name of the thing to remove from the group.
thingArn	ThingArn	no	The ARN of the thing to remove from the group.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  remove-thing-from-thing-group \
    [--thing-group-name <value>] \
    [--thing-group-arn <value>] \
    [--thing-name <value>] \
    [--thing-arn <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingGroupName": "string",
  "thingGroupArn": "string",
  "thingName": "string",
  "thingArn": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
thingGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The group name.
thingGroupArn	string	The group ARN.
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing to remove from the group.
thingArn	string	The ARN of the thing to remove from the group.

Output:

None

ReplaceTopicRule

Replaces the rule. You must specify all parameters for the new rule. Creating rules is an administrator-level action. Any user who has permission to create rules will be able to access data processed by the rule.

Request syntax:

```
PATCH /rules/ruleName
Content-type: application/json

{
  "topicRulePayload": {
    "sql": "string",
    "description": "string",
    "actions": [
      {
        "dynamoDB": {
          "tableName": "string",
          "roleArn": "string",
          "operation": "string",
          "hashKeyField": "string",
          "hashKeyValue": "string",
          "hashKeyType": "string",
          "rangeKeyField": "string",
          "rangeKeyValue": "string",
          "rangeKeyType": "string",
          "payloadField": "string"
        },
        "dynamoDBv2": {
          "roleArn": "string",
          "putItem": {
            "tableName": "string"
          }
        },
        "lambda": {
          "functionArn": "string"
        },
        "sns": {
          "targetArn": "string",
          "roleArn": "string",
          "messageFormat": "string"
        },
        "sqs": {
          "roleArn": "string",
          "queueUrl": "string",
          "useBase64": "boolean"
        },
        "kinesis": {
          "roleArn": "string",
          "streamName": "string",
          "partitionKey": "string"
        },
        "republish": {
          "roleArn": "string",
          "topic": "string"
        },
        "s3": {
          "roleArn": "string",
          "bucketName": "string",
```

```
        "key": "string",
        "cannedAcl": "string"
      },
      "firehose": {
        "roleArn": "string",
        "deliveryStreamName": "string",
        "separator": "string"
      },
      "cloudwatchMetric": {
        "roleArn": "string",
        "metricNamespace": "string",
        "metricName": "string",
        "metricValue": "string",
        "metricUnit": "string",
        "metricTimestamp": "string"
      },
      "cloudwatchAlarm": {
        "roleArn": "string",
        "alarmName": "string",
        "stateReason": "string",
        "stateValue": "string"
      },
      "elasticsearch": {
        "roleArn": "string",
        "endpoint": "string",
        "index": "string",
        "type": "string",
        "id": "string"
      },
      "salesforce": {
        "token": "string",
        "url": "string"
      }
    }
  ],
  "ruleDisabled": "boolean",
  "awsIotSqlVersion": "string",
  "errorAction": {
    "dynamoDB": {
      "tableName": "string",
      "roleArn": "string",
      "operation": "string",
      "hashKeyField": "string",
      "hashKeyValue": "string",
      "hashKeyType": "string",
      "rangeKeyField": "string",
      "rangeKeyValue": "string",
      "rangeKeyType": "string",
      "payloadField": "string"
    },
    "dynamoDBv2": {
      "roleArn": "string",
      "putItem": {
        "tableName": "string"
      }
    },
    "lambda": {
      "functionArn": "string"
    },
    "sns": {
      "targetArn": "string",
      "roleArn": "string",
      "messageFormat": "string"
    },
    "sqs": {
      "roleArn": "string",
```

```
        "queueUrl": "string",
        "useBase64": "boolean"
      },
      "kinesis": {
        "roleArn": "string",
        "streamName": "string",
        "partitionKey": "string"
      },
      "republish": {
        "roleArn": "string",
        "topic": "string"
      },
      "s3": {
        "roleArn": "string",
        "bucketName": "string",
        "key": "string",
        "cannedAcl": "string"
      },
      "firehose": {
        "roleArn": "string",
        "deliveryStreamName": "string",
        "separator": "string"
      },
      "cloudwatchMetric": {
        "roleArn": "string",
        "metricNamespace": "string",
        "metricName": "string",
        "metricValue": "string",
        "metricUnit": "string",
        "metricTimestamp": "string"
      },
      "cloudwatchAlarm": {
        "roleArn": "string",
        "alarmName": "string",
        "stateReason": "string",
        "stateValue": "string"
      },
      "elasticsearch": {
        "roleArn": "string",
        "endpoint": "string",
        "index": "string",
        "type": "string",
        "id": "string"
      },
      "salesforce": {
        "token": "string",
        "url": "string"
      }
    }
  }
}
```

URI Request Parameters:

Name	Type	Req?	Description
ruleName	RuleName	yes	The name of the rule.

Request Body Parameters:

Name	Type	Req?	Description
topicRulePayload	TopicRulePayload	yes	The rule payload.

Errors:

`SqlParseException`

The Rule-SQL expression can't be parsed correctly.

HTTP response code: 400

`InternalException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

CLI

Synopsis:

```
aws iot  replace-topic-rule \
    --rule-name <value> \
    --topic-rule-payload <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "ruleName": "string",
  "topicRulePayload": {
    "sql": "string",
    "description": "string",
    "actions": [
      {
        "dynamoDB": {
          "tableName": "string",
          "roleArn": "string",
          "operation": "string",
          "hashKeyField": "string",
          "hashKeyValue": "string",
          "hashKeyType": "string",
          "rangeKeyField": "string",
          "rangeKeyValue": "string",
          "rangeKeyType": "string",
          "payloadField": "string"
        },
```

```
      "dynamoDBv2": {
        "roleArn": "string",
        "putItem": {
          "tableName": "string"
        }
      },
      "lambda": {
        "functionArn": "string"
      },
      "sns": {
        "targetArn": "string",
        "roleArn": "string",
        "messageFormat": "string"
      },
      "sqs": {
        "roleArn": "string",
        "queueUrl": "string",
        "useBase64": "boolean"
      },
      "kinesis": {
        "roleArn": "string",
        "streamName": "string",
        "partitionKey": "string"
      },
      "republish": {
        "roleArn": "string",
        "topic": "string"
      },
      "s3": {
        "roleArn": "string",
        "bucketName": "string",
        "key": "string",
        "cannedAcl": "string"
      },
      "firehose": {
        "roleArn": "string",
        "deliveryStreamName": "string",
        "separator": "string"
      },
      "cloudwatchMetric": {
        "roleArn": "string",
        "metricNamespace": "string",
        "metricName": "string",
        "metricValue": "string",
        "metricUnit": "string",
        "metricTimestamp": "string"
      },
      "cloudwatchAlarm": {
        "roleArn": "string",
        "alarmName": "string",
        "stateReason": "string",
        "stateValue": "string"
      },
      "elasticsearch": {
        "roleArn": "string",
        "endpoint": "string",
        "index": "string",
        "type": "string",
        "id": "string"
      },
      "salesforce": {
        "token": "string",
        "url": "string"
      }
    }
  ],
```

```
  "ruleDisabled": "boolean",
  "awsIotSqlVersion": "string",
  "errorAction": {
    "dynamoDB": {
      "tableName": "string",
      "roleArn": "string",
      "operation": "string",
      "hashKeyField": "string",
      "hashKeyValue": "string",
      "hashKeyType": "string",
      "rangeKeyField": "string",
      "rangeKeyValue": "string",
      "rangeKeyType": "string",
      "payloadField": "string"
    },
    "dynamoDBv2": {
      "roleArn": "string",
      "putItem": {
        "tableName": "string"
      }
    },
    "lambda": {
      "functionArn": "string"
    },
    "sns": {
      "targetArn": "string",
      "roleArn": "string",
      "messageFormat": "string"
    },
    "sqs": {
      "roleArn": "string",
      "queueUrl": "string",
      "useBase64": "boolean"
    },
    "kinesis": {
      "roleArn": "string",
      "streamName": "string",
      "partitionKey": "string"
    },
    "republish": {
      "roleArn": "string",
      "topic": "string"
    },
    "s3": {
      "roleArn": "string",
      "bucketName": "string",
      "key": "string",
      "cannedAcl": "string"
    },
    "firehose": {
      "roleArn": "string",
      "deliveryStreamName": "string",
      "separator": "string"
    },
    "cloudwatchMetric": {
      "roleArn": "string",
      "metricNamespace": "string",
      "metricName": "string",
      "metricValue": "string",
      "metricUnit": "string",
      "metricTimestamp": "string"
    },
    "cloudwatchAlarm": {
      "roleArn": "string",
      "alarmName": "string",
      "stateReason": "string",
```

```
        "stateValue": "string"
      },
      "elasticsearch": {
        "roleArn": "string",
        "endpoint": "string",
        "index": "string",
        "type": "string",
        "id": "string"
      },
      "salesforce": {
        "token": "string",
        "url": "string"
      }
    }
  }
}
```

`cli-input-json` fields:

Name	Type	Description
ruleName	string length max:128 min:1 pattern: ^[a-zA-Z0-9_]+$	The name of the rule.
topicRulePayload	TopicRulePayload	The rule payload.
sql	string	The SQL statement used to query the topic. For more information, see AWS IoT SQL Reference in the *AWS IoT Developer Guide*.
description	string	The description of the rule.
actions	list member: Action	The actions associated with the rule.
Action	Action	
dynamoDB	DynamoDBAction	Write to a DynamoDB table.
tableName	string	The name of the DynamoDB table.
roleArn	string	The ARN of the IAM role that grants access to the DynamoDB table.
operation	string	The type of operation to be performed. This follows the substitution template, so it can be `$ operation`, but the substitution must result in one of the following: `INSERT`, `UPDATE`, or `DELETE`.
hashKeyField	string	The hash key name.

Name	Type	Description
hashKeyValue	string	The hash key value.
hashKeyType	string enum: STRING | NUMBER java class: iot.goldeneye.service.DynamoKeyType	The hash key type. Valid values are "STRING" or "NUMBER"
rangeKeyField	string	The range key name.
rangeKeyValue	string	The range key value.
rangeKeyType	string enum: STRING | NUMBER java class: iot.goldeneye.service.DynamoKeyType	The range key type. Valid values are "STRING" or "NUMBER"
payloadField	string	The action payload. This name can be customized.
dynamoDBv2	DynamoDBv2Action	Write to a DynamoDB table. This is a new version of the DynamoDB action. It allows you to write each attribute in an MQTT message payload into a separate DynamoDB column.
roleArn	string	The ARN of the IAM role that grants access to the DynamoDB table.
putItem	PutItemInput	Specifies the DynamoDB table to which the message data will be written. For example: `{ "dynamoDBv2": { "roleArn": "aws:iam:12341251:my-role" "putItem": { "tableName": "my-table" } } }` Each attribute in the message payload will be written to a separate column in the DynamoDB database.
tableName	string	The table where the message data will be written
lambda	LambdaAction	Invoke a Lambda function.
functionArn	string	The ARN of the Lambda function.

Name	Type	Description
sns	SnsAction	Publish to an Amazon SNS topic.
targetArn	string	The ARN of the SNS topic.
roleArn	string	The ARN of the IAM role that grants access.
messageFormat	string enum: RAW | JSON java class: iot.goldeneye.service.MessageFormat	The message format of the message to publish. Optional. Accepted values are "JSON" and "RAW". The default value of the attribute is "RAW". SNS uses this setting to determine if the payload should be parsed and relevant platform-specific bits of the payload should be extracted. To read more about SNS message formats, see http://docs.aws.amazon.com/sns/latest/dg/json-formats.html refer to their official documentation.
sqs	SqsAction	Publish to an Amazon SQS queue.
roleArn	string	The ARN of the IAM role that grants access.
queueUrl	string	The URL of the Amazon SQS queue.
useBase64	boolean java class: java.lang.Boolean	Specifies whether to use Base64 encoding.
kinesis	KinesisAction	Write data to an Amazon Kinesis stream.
roleArn	string	The ARN of the IAM role that grants access to the Amazon Kinesis stream.
streamName	string	The name of the Amazon Kinesis stream.
partitionKey	string	The partition key.
republish	RepublishAction	Publish to another MQTT topic.
roleArn	string	The ARN of the IAM role that grants access.
topic	string	The name of the MQTT topic.
s3	S3Action	Write to an Amazon S3 bucket.

Name	Type	Description
roleArn	string	The ARN of the IAM role that grants access.
bucketName	string	The Amazon S3 bucket.
key	string	The object key.
cannedAcl	string enum: private | public-read | public-read-write | aws-exec-read | authenticated-read | bucket-owner-read | bucket-owner-full-control | log-delivery-write java class: iot.goldeneye.service.CannedAccessControlList	The Amazon S3 canned ACL that controls access to the object identified by the object key. For more information, see S3 canned ACLs.
firehose	FirehoseAction	Write to an Amazon Kinesis Firehose stream.
roleArn	string	The IAM role that grants access to the Amazon Kinesis Firehose stream.
deliveryStreamName	string	The delivery stream name.
separator	string pattern: ([ ])|( )|(,)	A character separator that will be used to separate records written to the Firehose stream. Valid values are: '\n' (newline), '\t' (tab), '\r\n' (Windows newline), ',' (comma).
cloudwatchMetric	CloudwatchMetricAction	Capture a CloudWatch metric.
roleArn	string	The IAM role that allows access to the CloudWatch metric.
metricNamespace	string	The CloudWatch metric namespace name.
metricName	string	The CloudWatch metric name.
metricValue	string	The CloudWatch metric value.
metricUnit	string	The metric unit supported by CloudWatch.
metricTimestamp	string	An optional Unix timestamp.
cloudwatchAlarm	CloudwatchAlarmAction	Change the state of a CloudWatch alarm.
roleArn	string	The IAM role that allows access to the CloudWatch alarm.
alarmName	string	The CloudWatch alarm name.

Name	Type	Description
stateReason	string	The reason for the alarm change.
stateValue	string	The value of the alarm state. Acceptable values are: OK, ALARM, INSUFFICIENT_DATA.
elasticsearch	ElasticsearchAction	Write data to an Amazon Elasticsearch Service domain.
roleArn	string	The IAM role ARN that has access to Elasticsearch.
endpoint	string pattern: https?://.*	The endpoint of your Elasticsearch domain.
index	string	The Elasticsearch index where you want to store your data.
type	string	The type of document you are storing.
id	string	The unique identifier for the document you are storing.
salesforce	SalesforceAction	Send a message to a Salesforce IoT Cloud Input Stream.
token	string length min:40	The token used to authenticate access to the Salesforce IoT Cloud Input Stream. The token is available from the Salesforce IoT Cloud platform after creation of the Input Stream.
url	string length max:2000 pattern: https://ingestion-[a-zA-Z0-9]{1,12}.[a-zA-Z0-9]+.((sfdc-matrix.net)|(sfdcnow.com))/streams/w *1,20*/w *1,20*/event	The URL exposed by the Salesforce IoT Cloud Input Stream. The URL is available from the Salesforce IoT Cloud platform after creation of the Input Stream.
ruleDisabled	boolean java class: java.lang.Boolean	Specifies whether the rule is disabled.
awsIotSqlVersion	string	The version of the SQL rules engine to use when evaluating the rule.
errorAction	Action	The action to take when an error occurs.
dynamoDB	DynamoDBAction	Write to a DynamoDB table.

Name	Type	Description
tableName	string	The name of the DynamoDB table.
roleArn	string	The ARN of the IAM role that grants access to the DynamoDB table.
operation	string	The type of operation to be performed. This follows the substitution template, so it can be `$ operation`, but the substitution must result in one of the following: `INSERT`, `UPDATE`, or `DELETE`.
hashKeyField	string	The hash key name.
hashKeyValue	string	The hash key value.
hashKeyType	string enum: STRING | NUMBER java class: iot.goldeneye.service.DynamoKeyType	The hash key type. Valid values are "STRING" or "NUMBER"
rangeKeyField	string	The range key name.
rangeKeyValue	string	The range key value.
rangeKeyType	string enum: STRING | NUMBER java class: iot.goldeneye.service.DynamoKeyType	The range key type. Valid values are "STRING" or "NUMBER"
payloadField	string	The action payload. This name can be customized.
dynamoDBv2	DynamoDBv2Action	Write to a DynamoDB table. This is a new version of the DynamoDB action. It allows you to write each attribute in an MQTT message payload into a separate DynamoDB column.
roleArn	string	The ARN of the IAM role that grants access to the DynamoDB table.

Name	Type	Description
putItem	PutItemInput	Specifies the DynamoDB table to which the message data will be written. For example: `{ "dynamoDBv2": { "roleArn": "aws:iam:12341251:my-role" "putItem": { "tableName": "my-table" } } }` Each attribute in the message payload will be written to a separate column in the DynamoDB database.
tableName	string	The table where the message data will be written
lambda	LambdaAction	Invoke a Lambda function.
functionArn	string	The ARN of the Lambda function.
sns	SnsAction	Publish to an Amazon SNS topic.
targetArn	string	The ARN of the SNS topic.
roleArn	string	The ARN of the IAM role that grants access.
messageFormat	string enum: RAW | JSON java class: iot.goldeneye.service.MessageFormat	The message format of the message to publish. Optional. Accepted values are "JSON" and "RAW". The default value of the attribute is "RAW". SNS uses this setting to determine if the payload should be parsed and relevant platform-specific bits of the payload should be extracted. To read more about SNS message formats, see http://docs.aws.amazon.com/sns/latest/dg/json-formats.html refer to their official documentation.
sqs	SqsAction	Publish to an Amazon SQS queue.
roleArn	string	The ARN of the IAM role that grants access.
queueUrl	string	The URL of the Amazon SQS queue.

Name	Type	Description
useBase64	boolean java class: java.lang.Boolean	Specifies whether to use Base64 encoding.
kinesis	KinesisAction	Write data to an Amazon Kinesis stream.
roleArn	string	The ARN of the IAM role that grants access to the Amazon Kinesis stream.
streamName	string	The name of the Amazon Kinesis stream.
partitionKey	string	The partition key.
republish	RepublishAction	Publish to another MQTT topic.
roleArn	string	The ARN of the IAM role that grants access.
topic	string	The name of the MQTT topic.
s3	S3Action	Write to an Amazon S3 bucket.
roleArn	string	The ARN of the IAM role that grants access.
bucketName	string	The Amazon S3 bucket.
key	string	The object key.
cannedAcl	string enum: private | public-read | public-read-write | aws-exec-read | authenticated-read | bucket-owner-read | bucket-owner-full-control | log-delivery-write java class: iot.goldeneye.service.CannedAccessControlList	The Amazon S3 canned ACL that controls access to the object identified by the object key. For more information, see S3 canned ACLs.
firehose	FirehoseAction	Write to an Amazon Kinesis Firehose stream.
roleArn	string	The IAM role that grants access to the Amazon Kinesis Firehose stream.
deliveryStreamName	string	The delivery stream name.
separator	string pattern: ([ ])|( )|(,)	A character separator that will be used to separate records written to the Firehose stream. Valid values are: '\n' (newline), '\t' (tab), '\r\n' (Windows newline), ',' (comma).

Name	Type	Description
cloudwatchMetric	CloudwatchMetricAction	Capture a CloudWatch metric.
roleArn	string	The IAM role that allows access to the CloudWatch metric.
metricNamespace	string	The CloudWatch metric namespace name.
metricName	string	The CloudWatch metric name.
metricValue	string	The CloudWatch metric value.
metricUnit	string	The metric unit supported by CloudWatch.
metricTimestamp	string	An optional Unix timestamp.
cloudwatchAlarm	CloudwatchAlarmAction	Change the state of a CloudWatch alarm.
roleArn	string	The IAM role that allows access to the CloudWatch alarm.
alarmName	string	The CloudWatch alarm name.
stateReason	string	The reason for the alarm change.
stateValue	string	The value of the alarm state. Acceptable values are: OK, ALARM, INSUFFICIENT_DATA.
elasticsearch	ElasticsearchAction	Write data to an Amazon Elasticsearch Service domain.
roleArn	string	The IAM role ARN that has access to Elasticsearch.
endpoint	string pattern: https?://.*	The endpoint of your Elasticsearch domain.
index	string	The Elasticsearch index where you want to store your data.
type	string	The type of document you are storing.
id	string	The unique identifier for the document you are storing.
salesforce	SalesforceAction	Send a message to a Salesforce IoT Cloud Input Stream.

Name	Type	Description
token	string length min:40	The token used to authenticate access to the Salesforce IoT Cloud Input Stream. The token is available from the Salesforce IoT Cloud platform after creation of the Input Stream.
url	string length max:2000 pattern: https://ingestion-[a-zA-Z0-9]{1,12}.[a-zA-Z0-9]+.((sfdc-matrix.net)|(sfdcnow.com))/streams/w *1,20*/w *1,20*/event	The URL exposed by the Salesforce IoT Cloud Input Stream. The URL is available from the Salesforce IoT Cloud platform after creation of the Input Stream.

Output:

None

SearchIndex

The query search index.

Request syntax:

```
POST /indices/search
Content-type: application/json

{
  "indexName": "string",
  "queryString": "string",
  "nextToken": "string",
  "maxResults": "integer",
  "queryVersion": "string"
}
```

Request Body Parameters:

Name	Type	Req?	Description
indexName	IndexName	no	The search index name.
queryString	QueryString	yes	The search query string.
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	QueryMaxResults	no	The maximum number of results to return at one time.
queryVersion	QueryVersion	no	The query version.

Response syntax:

```
Content-type: application/json

{
  "nextToken": "string",
  "things": [
    {
      "thingName": "string",
      "thingId": "string",
      "thingTypeName": "string",
      "thingGroupNames": [
        "string"
      ],
      "attributes": {
        "string": "string"
      },
      "shadow": "string"
    }
  ]
}
```

Response Body Parameters:

Name	Type	Req?	Description
nextToken	NextToken	no	The token used to get the next set of results, or **null** if there are no additional results.
things	ThingDocumentList	no	The things that match the search query.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidQueryException`

The query is invalid.

HTTP response code: 400

`IndexNotReadyException`

The index is not ready.

HTTP response code: 400

CLI

Synopsis:

```
aws iot  search-index \
    [--index-name <value>] \
    --query-string <value> \
    [--next-token <value>] \
    [--max-results <value>] \
    [--query-version <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "indexName": "string",
  "queryString": "string",
  "nextToken": "string",
  "maxResults": "integer",
  "queryVersion": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
indexName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The search index name.
queryString	string length max:1000 min:1	The search query string.

Name	Type	Description
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.
maxResults	integer java class: java.lang.Integer range- max:500 min:1	The maximum number of results to return at one time.
queryVersion	string	The query version.

Output:

```
{
  "nextToken": "string",
  "things": [
    {
      "thingName": "string",
      "thingId": "string",
      "thingTypeName": "string",
      "thingGroupNames": [
        "string"
      ],
      "attributes": {
        "string": "string"
      },
      "shadow": "string"
    }
  ]
}
```

cli output fields:

Name	Type	Description
nextToken	string	The token used to get the next set of results, or **null** if there are no additional results.
things	list member: ThingDocument java class: java.util.List	The things that match the search query.
ThingDocument	ThingDocument	
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The thing name.
thingId	string	The thing ID.
thingTypeName	string length max:128 min:1	The thing type name.

Name	Type	Description
	pattern: [a-zA-Z0-9:_-]+	
thingGroupNames	list member: ThingGroupName java class: java.util.List	Thing group names.
ThingGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
attributes	map key: AttributeName value: AttributeValue	The attributes.
AttributeName	string length max:128 pattern: [a-zA-Z0-9_.,@/:#-]+	
AttributeValue	string length max:800 pattern: [a-zA-Z0-9_.,@/:#-]*	
shadow	string	The thing shadow.

SetDefaultAuthorizer

Sets the default authorizer. This will be used if a websocket connection is made without specifying an authorizer.

Request syntax:

```
POST /default-authorizer
Content-type: application/json

{
  "authorizerName": "string"
}
```

Request Body Parameters:

Name	Type	Req?	Description
authorizerName	AuthorizerName	yes	The authorizer name.

Response syntax:

```
Content-type: application/json

{
  "authorizerName": "string",
  "authorizerArn": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
authorizerName	AuthorizerName	no	The authorizer name.
authorizerArn	AuthorizerArn	no	The authorizer ARN.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  set-default-authorizer \
```

```
    --authorizer-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "authorizerName": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
authorizerName	string length max:128 min:1 pattern: [w=,@-]+	The authorizer name.

Output:

```
{
  "authorizerName": "string",
  "authorizerArn": "string"
}
```

cli output fields:

Name	Type	Description
authorizerName	string length max:128 min:1 pattern: [w=,@-]+	The authorizer name.
authorizerArn	string	The authorizer ARN.

SetDefaultPolicyVersion

Sets the specified version of the specified policy as the policy's default (operative) version. This action affects all certificates to which the policy is attached. To list the principals the policy is attached to, use the ListPrincipalPolicy API.

Request syntax:

```
PATCH /policies/policyName/version/policyVersionId
```

URI Request Parameters:

Name	Type	Req?	Description
policyName	PolicyName	yes	The policy name.

Name	Type	Req?	Description
policyVersionId	PolicyVersionId	yes	The policy version ID.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  set-default-policy-version \
    --policy-name <value> \
    --policy-version-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "policyName": "string",
  "policyVersionId": "string"
```

```
}
```

`cli-input-json` fields:

Name	Type	Description
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.
policyVersionId	string pattern: [0-9]+	The policy version ID.

Output:

None

SetLoggingOptions

Sets the logging options.

Request syntax:

```
POST /loggingOptions
Content-type: application/json

{
  "loggingOptionsPayload": {
    "roleArn": "string",
    "logLevel": "string"
  }
}
```

Request Body Parameters:

Name	Type	Req?	Description
loggingOptionsPayload	LoggingOptionsPayload	yes	The logging options payload.

Errors:

`InternalException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  set-logging-options \
    --logging-options-payload <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "loggingOptionsPayload": {
    "roleArn": "string",
    "logLevel": "string"
  }
}
```

`cli-input-json` fields:

Name	Type	Description
loggingOptionsPayload	LoggingOptionsPayload	The logging options payload.
roleArn	string	The ARN of the IAM role that grants access.
logLevel	string enum: DEBUG | INFO | ERROR | WARN | DISABLED java class: iot.goldeneye.service.LogLevel	The log level.

Output:

None

SetV2LoggingLevel

Sets the logging level.

Request syntax:

```
POST /v2LoggingLevel
```

```
Content-type: application/json

{
  "logTarget": {
    "targetType": "string",
    "targetName": "string"
  },
  "logLevel": "string"
}
```

Request Body Parameters:

Name	Type	Req?	Description
logTarget	LogTarget	yes	The log target.
logLevel	LogLevel	yes	The log level.

Errors:

`InternalException`

An unexpected error has occurred.

HTTP response code: 500

`NotConfiguredException`

The resource is not configured.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  set-v2-logging-level \
    --log-target <value> \
    --log-level <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
```

```
  "logTarget": {
    "targetType": "string",
    "targetName": "string"
  },
  "logLevel": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
logTarget	LogTarget	The log target.
targetType	string enum: DEFAULT | THING_GROUP java class: iot.goldeneye.service.LogTargetType	The target type.
targetName	string	The target name.
logLevel	string enum: DEBUG | INFO | ERROR | WARN | DISABLED java class: iot.goldeneye.service.LogLevel	The log level.

Output:

None

SetV2LoggingOptions

Sets the logging options for the V2 logging service.

Request syntax:

```
POST /v2LoggingOptions
Content-type: application/json

{
  "roleArn": "string",
  "defaultLogLevel": "string",
  "disableAllLogs": "boolean"
}
```

Request Body Parameters:

Name	Type	Req?	Description
roleArn	AwsArn	no	The role ARN that allows IoT to write to Cloudwatch logs.

Name	Type	Req?	Description
defaultLogLevel	LogLevel	no	The default logging level.
disableAllLogs	DisableAllLogs	no	Set to true to disable all logs, otherwise set to false.

Errors:

`InternalException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

CLI

Synopsis:

```
aws iot  set-v2-logging-options \
    [--role-arn <value>] \
    [--default-log-level <value>] \
    [--disable-all-logs | --no-disable-all-logs]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "roleArn": "string",
  "defaultLogLevel": "string",
  "disableAllLogs": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
roleArn	string	The role ARN that allows IoT to write to Cloudwatch logs.

Name	Type	Description
defaultLogLevel	string enum: DEBUG | INFO | ERROR | WARN | DISABLED java class: iot.goldeneye.service.LogLevel	The default logging level.
disableAllLogs	boolean	Set to true to disable all logs, otherwise set to false.

Output:

None

StartNextPendingJobExecution

Gets and starts the next pending (status IN_PROGRESS or QUEUED) job execution for a thing.

Request syntax:

```
PUT /things/thingName/jobs/$next
Content-type: application/json

{
  "statusDetails": {
    "string": "string"
  }
}
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	yes	The name of the thing associated with the device.

Request Body Parameters:

Name	Type	Req?	Description
statusDetails	DetailsMap	no	A collection of name/value pairs that describe the status of the job execution. If not specified, the statusDetails are unchanged.

Response syntax:

```
Content-type: application/json

{
  "execution": {
    "jobId": "string",
    "thingName": "string",
    "status": "string",
    "statusDetails": {
      "string": "string"
    },
    "queuedAt": "long",
    "startedAt": "long",
    "lastUpdatedAt": "long",
    "versionNumber": "long",
    "executionNumber": "long",
    "jobDocument": "string"
  }
}
```

Response Body Parameters:

Name	Type	Req?	Description
execution	JobExecution	no	A JobExecution object.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`CertificateValidationException`

The certificate is invalid.

HTTP response code: 400

CLI

Synopsis:

```
aws iot  start-next-pending-job-execution \
    --thing-name <value> \
    [--status-details <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string",
  "statusDetails": {
    "string": "string"
  }
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing associated with the device.
statusDetails	map key: DetailsKey value: DetailsValue	A collection of name/value pairs that describe the status of the job execution. If not specified, the statusDetails are unchanged.
DetailsKey	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
DetailsValue	string length max:1024 min:1 pattern: [^\\p{C}]*+	

Output:

```
{
  "execution": {
    "jobId": "string",
    "thingName": "string",
    "status": "string",
    "statusDetails": {
      "string": "string"
    },
    "queuedAt": "long",
    "startedAt": "long",
    "lastUpdatedAt": "long",
    "versionNumber": "long",
    "executionNumber": "long",
    "jobDocument": "string"
```

```
    }
}
```

cli output fields:

Name	Type	Description
execution	JobExecution	A JobExecution object.
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier you assigned to this job when it was created.
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing that is executing the job.
status	string enum: QUEUED | IN_PROGRESS | SUCCEEDED | FAILED | REJECTED | REMOVED | CANCELED java class: com.amazonaws.iot.laser.common.JobExecutionStatus	The status of the job execution. Can be one of: "QUEUED", "IN_PROGRESS", "FAILED", "SUCCESS", "CANCELED", "REJECTED", or "REMOVED".
statusDetails	map key: DetailsKey value: DetailsValue	A collection of name/value pairs that describe the status of the job execution.
DetailsKey	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
DetailsValue	string length max:1024 min:1 pattern: [^\\p{C}]*+	
queuedAt	long	The time, in milliseconds since the epoch, when the job execution was enqueued.
startedAt	long java class: java.lang.Long	The time, in milliseconds since the epoch, when the job execution was started.
lastUpdatedAt	long	The time, in milliseconds since the epoch, when the job execution was last updated.
versionNumber	long	The version of the job execution. Job execution versions are

Name	Type	Description
		incremented each time they are updated by a device.
executionNumber	long java class: java.lang.Long	A number that identifies a particular job execution on a particular device. It can be used later in commands that return or update job execution information.
jobDocument	string length max:32768	The content of the job document.

StartThingRegistrationTask

Creates a bulk thing provisioning task.

Request syntax:

```
POST /thing-registration-tasks
Content-type: application/json

{
  "templateBody": "string",
  "inputFileBucket": "string",
  "inputFileKey": "string",
  "roleArn": "string"
}
```

Request Body Parameters:

Name	Type	Req?	Description
templateBody	TemplateBody	yes	The provisioning template.
inputFileBucket	RegistryS3BucketName	yes	The S3 bucket that contains the input file.
inputFileKey	RegistryS3KeyName	yes	The name of input file within the S3 bucket. This file contains a newline delimited JSON file. Each line contains the parameter values to provision one device (thing).
roleArn	RoleArn	yes	The IAM role ARN that grants permission the input file.

Response syntax:

```
Content-type: application/json

{
  "taskId": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
taskId	TaskId	no	The bulk thing provisioning task ID.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  start-thing-registration-task \
    --template-body <value> \
    --input-file-bucket <value> \
    --input-file-key <value> \
    --role-arn <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "templateBody": "string",
  "inputFileBucket": "string",
```

```
  "inputFileKey": "string",
  "roleArn": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
templateBody	string	The provisioning template.
inputFileBucket	string length max:256 min:3 pattern: [a-zA-Z0-9._-]+	The S3 bucket that contains the input file.
inputFileKey	string length max:1024 min:1 pattern: [a-zA-Z0-9!_.*'()-/]+	The name of input file within the S3 bucket. This file contains a newline delimited JSON file. Each line contains the parameter values to provision one device (thing).
roleArn	string length max:2048 min:20	The IAM role ARN that grants permission the input file.

Output:

```
{
  "taskId": "string"
}
```

cli output fields:

Name	Type	Description
taskId	string length max:40	The bulk thing provisioning task ID.

StopThingRegistrationTask

Cancels a bulk thing provisioning task.

Request syntax:

```
PUT /thing-registration-tasks/taskId/cancel
```

URI Request Parameters:

Name	Type	Req?	Description
taskId	TaskId	yes	The bulk thing provisioning task ID.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  stop-thing-registration-task \
    --task-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "taskId": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
taskId	string length max:40	The bulk thing provisioning task ID.

Output:

None

TestAuthorization

Test custom authorization.

Request syntax:

```
POST /test-authorization?clientId=clientId
Content-type: application/json

{
  "principal": "string",
  "cognitoIdentityPoolId": "string",
  "authInfos": [
    {
      "actionType": "string",
      "resources": [
        "string"
      ]
    }
  ],
  "policyNamesToAdd": [
    "string"
  ],
  "policyNamesToSkip": [
    "string"
  ]
}
```

URI Request Parameters:

Name	Type	Req?	Description
clientId	ClientId	no	The MQTT client ID.

Request Body Parameters:

Name	Type	Req?	Description
principal	Principal	no	The principal.
cognitoIdentityPoolId	CognitoIdentityPoolId	no	The Cognito identity pool ID.
authInfos	AuthInfos	yes	A list of authorization info objects. Simulating authorization will create a response for each `authInfo` object in the list.
policyNamesToAdd	PolicyNames	no	When testing custom authorization, the policies specified here are treated as if they are attached to

Name	Type	Req?	Description
			the principal being authorized.
policyNamesToSkip	PolicyNames	no	When testing custom authorization, the policies specified here are treated as if they are not attached to the principal being authorized.

Response syntax:

```
Content-type: application/json

{
  "authResults": [
    {
      "authInfo": {
        "actionType": "string",
        "resources": [
          "string"
        ]
      },
      "allowed": {
        "policies": [
          {
            "policyName": "string",
            "policyArn": "string"
          }
        ]
      },
      "denied": {
        "implicitDeny": {
          "policies": [
            {
              "policyName": "string",
              "policyArn": "string"
            }
          ]
        },
        "explicitDeny": {
          "policies": [
            {
              "policyName": "string",
              "policyArn": "string"
            }
          ]
        }
      },
      "authDecision": "string",
      "missingContextValues": [
        "string"
      ]
    }
  ]
}
```

Response Body Parameters:

Name	Type	Req?	Description
authResults	AuthResults	no	The authentication results.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`LimitExceededException`

The number of attached entities exceeds the limit.

HTTP response code: 410

CLI

Synopsis:

```
aws iot  test-authorization \
    [--principal <value>] \
    [--cognito-identity-pool-id <value>] \
    --auth-infos <value> \
    [--client-id <value>] \
    [--policy-names-to-add <value>] \
    [--policy-names-to-skip <value>]  \
    [--cli-input-json <value>] \
```

```
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "principal": "string",
  "cognitoIdentityPoolId": "string",
  "authInfos": [
    {
      "actionType": "string",
      "resources": [
        "string"
      ]
    }
  ],
  "clientId": "string",
  "policyNamesToAdd": [
    "string"
  ],
  "policyNamesToSkip": [
    "string"
  ]
}
```

`cli-input-json` fields:

Name	Type	Description
principal	string	The principal.
cognitoIdentityPoolId	string	The Cognito identity pool ID.
authInfos	list member: AuthInfo	A list of authorization info objects. Simulating authorization will create a response for each `authInfo` object in the list.
AuthInfo	AuthInfo	
actionType	string enum: PUBLISH | SUBSCRIBE | RECEIVE | CONNECT java class: com.amazonaws.iot.identity.enums.ActionType	The type of action for which the principal is being authorized.
resources	list member: Resource	The resources for which the principal is being authorized to perform the specified action.
Resource	string	
clientId	string	The MQTT client ID.
policyNamesToAdd	list member: PolicyName java class: java.util.List	When testing custom authorization, the policies specified here are treated as if they are attached to the principal being authorized.

Name	Type	Description
PolicyName	string length max:128 min:1 pattern: [w+=,.@-]+	
policyNamesToSkip	list member: PolicyName java class: java.util.List	When testing custom authorization, the policies specified here are treated as if they are not attached to the principal being authorized.
PolicyName	string length max:128 min:1 pattern: [w+=,.@-]+	

Output:

```
{
  "authResults": [
    {
      "authInfo": {
        "actionType": "string",
        "resources": [
          "string"
        ]
      },
      "allowed": {
        "policies": [
          {
            "policyName": "string",
            "policyArn": "string"
          }
        ]
      },
      "denied": {
        "implicitDeny": {
          "policies": [
            {
              "policyName": "string",
              "policyArn": "string"
            }
          ]
        },
        "explicitDeny": {
          "policies": [
            {
              "policyName": "string",
              "policyArn": "string"
            }
          ]
        }
      },
      "authDecision": "string",
      "missingContextValues": [
        "string"
      ]
    }
```

```
  ]
}
```

cli output fields:

Name	Type	Description
authResults	list member: AuthResult	The authentication results.
AuthResult	AuthResult	
authInfo	AuthInfo	Authorization information.
actionType	string enum: PUBLISH | SUBSCRIBE | RECEIVE | CONNECT java class: com.amazonaws.iot.identity.enums.ActionType	The type of action for which the principal is being authorized.
resources	list member: Resource	The resources for which the principal is being authorized to perform the specified action.
Resource	string	
allowed	Allowed	The policies and statements that allowed the specified action.
policies	list member: Policy java class: java.util.List	A list of policies that allowed the authentication.
Policy	Policy	
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.
policyArn	string	The policy ARN.
denied	Denied	The policies and statements that denied the specified action.
implicitDeny	ImplicitDeny	Information that implicitly denies the authorization. When a policy doesn't explicitly deny or allow an action on a resource it is considered an implicit deny.
policies	list member: Policy	Policies that don't contain a matching allow or deny statement for the specified action on the specified resource.

Name	Type	Description
	java class: java.util.List	
Policy	Policy	
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.
policyArn	string	The policy ARN.
explicitDeny	ExplicitDeny	Information that explicitly denies the authorization.
policies	list member: Policy java class: java.util.List	The policies that denied the authorization.
Policy	Policy	
policyName	string length max:128 min:1 pattern: [w+=,.@-]+	The policy name.
policyArn	string	The policy ARN.
authDecision	string enum: ALLOWED | EXPLICIT_DENY | IMPLICIT_DENY java class: com.amazonaws.iot.identity.enums.AuthDecision	The final authorization decision of this scenario. Multiple statements are taken into account when determining the authorization decision. An explicit deny statement can override multiple allow statements.
missingContextValues	list member: MissingContextValue java class: java.util.List	Contains any missing context values found while evaluating policy.
MissingContextValue	string	

TestInvokeAuthorizer

Invoke the specified custom authorizer for testing purposes.

Request syntax:

```
POST /authorizer/authorizerName/test
Content-type: application/json

{
```

```
  "token": "string",
  "tokenSignature": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
authorizerName	AuthorizerName	yes	The custom authorizer name.

Request Body Parameters:

Name	Type	Req?	Description
token	Token	yes	The token returned by your custom authentication service.
tokenSignature	TokenSignature	yes	The signature made with the token and your custom authentication service's private key.

Response syntax:

```
Content-type: application/json

{
  "isAuthenticated": "boolean",
  "principalId": "string",
  "policyDocuments": [
    "string"
  ],
  "refreshAfterInSeconds": "integer",
  "disconnectAfterInSeconds": "integer"
}
```

Response Body Parameters:

Name	Type	Req?	Description
isAuthenticated	IsAuthenticated	no	True if the token is authenticated, otherwise false.
principalId	PrincipalId	no	The principal ID.
policyDocuments	PolicyDocuments	no	IAM policy documents.
refreshAfterInSeconds	Seconds	no	The number of seconds after which the temporary credentials are refreshed.
disconnectAfterInSeconds	Seconds	no	The number of seconds after which

Name	Type	Req?	Description
			the connection is terminated.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`InvalidResponseException`

The response is invalid.

HTTP response code: 400

CLI

Synopsis:

```
aws iot  test-invoke-authorizer \
    --authorizer-name <value> \
    --token <value> \
    --token-signature <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "authorizerName": "string",
  "token": "string",
  "tokenSignature": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
authorizerName	string length max:128 min:1 pattern: [w=,@-]+	The custom authorizer name.
token	string length max:1024 min:1	The token returned by your custom authentication service.
tokenSignature	string length max:2560 min:1 pattern: [A-Za-z0-9+/]+={0,2}	The signature made with the token and your custom authentication service's private key.

Output:

```
{
  "isAuthenticated": "boolean",
  "principalId": "string",
  "policyDocuments": [
    "string"
  ],
  "refreshAfterInSeconds": "integer",
  "disconnectAfterInSeconds": "integer"
}
```

cli output fields:

Name	Type	Description
isAuthenticated	boolean java class: java.lang.Boolean	True if the token is authenticated, otherwise false.
principalId	string length max:128 min:1 pattern: [a-zA-Z0-9]+	The principal ID.
policyDocuments	list member: PolicyDocument	IAM policy documents.
PolicyDocument	string	

Name	Type	Description
refreshAfterInSeconds	integer java class: java.lang.Integer	The number of seconds after which the temporary credentials are refreshed.
disconnectAfterInSeconds	integer java class: java.lang.Integer	The number of seconds after which the connection is terminated.

TransferCertificate

Transfers the specified certificate to the specified AWS account.

You can cancel the transfer until it is acknowledged by the recipient.

No notification is sent to the transfer destination's account. It is up to the caller to notify the transfer target.

The certificate being transferred must not be in the ACTIVE state. You can use the UpdateCertificate API to deactivate it.

The certificate must not have any policies attached to it. You can use the DetachPrincipalPolicy API to detach them.

Request syntax:

```
PATCH /transfer-certificate/certificateId?targetAwsAccount=targetAwsAccount
Content-type: application/json

{
  "transferMessage": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
certificateId	CertificateId	yes	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)
targetAwsAccount	AwsAccountId	yes	The AWS account.

Request Body Parameters:

Name	Type	Req?	Description
transferMessage	Message	no	The transfer message.

Response syntax:

```
Content-type: application/json
```

```
{
  "transferredCertificateArn": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
transferredCertificateArn	CertificateArn	no	The ARN of the certificate.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`CertificateStateException`

The certificate operation is not allowed.

HTTP response code: 406

`TransferConflictException`

You can't transfer the certificate because authorization policies are still attached.

HTTP response code: 409

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  transfer-certificate \
    --certificate-id <value> \
    --target-aws-account <value> \
    [--transfer-message <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "certificateId": "string",
  "targetAwsAccount": "string",
  "transferMessage": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)
targetAwsAccount	string pattern: [0-9]{12}	The AWS account.
transferMessage	string length max:128	The transfer message.

Output:

```
{
  "transferredCertificateArn": "string"
}
```

cli output fields:

Name	Type	Description
transferredCertificateArn	string	The ARN of the certificate.

UpdateAuthorizer

Updates an authorizer.

Request syntax:

```
PUT /authorizer/authorizerName
Content-type: application/json

{
```

```
  "authorizerFunctionArn": "string",
  "tokenKeyName": "string",
  "tokenSigningPublicKeys": {
    "string": "string"
  },
  "status": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
authorizerName	AuthorizerName	yes	The authorizer name.

Request Body Parameters:

Name	Type	Req?	Description
authorizerFunctionArn	AuthorizerFunctionArn	no	The ARN of the authorizer's Lambda function.
tokenKeyName	TokenKeyName	no	The key used to extract the token from the HTTP headers.
tokenSigningPublicKeys	PublicKeyMap	no	The public keys used to verify the token signature.
status	AuthorizerStatus	no	The status of the update authorizer request.

Response syntax:

```
Content-type: application/json

{
  "authorizerName": "string",
  "authorizerArn": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
authorizerName	AuthorizerName	no	The authorizer name.
authorizerArn	AuthorizerArn	no	The authorizer ARN.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`LimitExceededException`

The number of attached entities exceeds the limit.

HTTP response code: 410

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  update-authorizer \
    --authorizer-name <value> \
    [--authorizer-function-arn <value>] \
    [--token-key-name <value>] \
    [--token-signing-public-keys <value>] \
    [--status <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "authorizerName": "string",
  "authorizerFunctionArn": "string",
  "tokenKeyName": "string",
  "tokenSigningPublicKeys": {
    "string": "string"
  },
  "status": "string"
```

```
}
```

`cli-input-json` fields:

Name	Type	Description
authorizerName	string length max:128 min:1 pattern: [w=,@-]+	The authorizer name.
authorizerFunctionArn	string	The ARN of the authorizer's Lambda function.
tokenKeyName	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The key used to extract the token from the HTTP headers.
tokenSigningPublicKeys	map key: KeyName value: KeyValue	The public keys used to verify the token signature.
KeyName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
KeyValue	string length max:5120	
status	string enum: ACTIVE | INACTIVE java class: iot.identity.service.AuthorizerStatus	The status of the update authorizer request.

Output:

```
{
  "authorizerName": "string",
  "authorizerArn": "string"
}
```

cli output fields:

Name	Type	Description
authorizerName	string length max:128 min:1 pattern: [w=,@-]+	The authorizer name.

Name	Type	Description
authorizerArn	string	The authorizer ARN.

UpdateCACertificate

Updates a registered CA certificate.

Request syntax:

```
PUT /cacertificate/caCertificateId?
newStatus=newStatus&newAutoRegistrationStatus=newAutoRegistrationStatus
Content-type: application/json

{
  "registrationConfig": {
    "templateBody": "string",
    "roleArn": "string"
  },
  "removeAutoRegistration": "boolean"
}
```

URI Request Parameters:

Name	Type	Req?	Description
certificateId	CertificateId	yes	The CA certificate identifier.
newStatus	CACertificateStatus	no	The updated status of the CA certificate. **Note:** The status value REGISTER_INACTIVE is deprecated and should not be used.
newAutoRegistrationStatus	AutoRegistrationStatus	no	The new value for the auto registration status. Valid values are: "ENABLE" or "DISABLE".

Request Body Parameters:

Name	Type	Req?	Description
registrationConfig	RegistrationConfig	no	Information about the registration configuration.
removeAutoRegistration	RemoveAutoRegistration	no	If true, remove auto registration.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  update-ca-certificate \
    --certificate-id <value> \
    [--new-status <value>] \
    [--new-auto-registration-status <value>] \
    [--registration-config <value>] \
    [--remove-auto-registration | --no-remove-auto-registration]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "certificateId": "string",
  "newStatus": "string",
  "newAutoRegistrationStatus": "string",
  "registrationConfig": {
    "templateBody": "string",
    "roleArn": "string"
```

```
  },
  "removeAutoRegistration": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The CA certificate identifier.
newStatus	string enum: ACTIVE | INACTIVE java class: iot.identity.service.CACertificateStatus	The updated status of the CA certificate. **Note:** The status value REGISTER_INACTIVE is deprecated and should not be used.
newAutoRegistrationStatus	string enum: ENABLE | DISABLE java class: iot.identity.service.AutoRegistrationStatus	The new value for the auto registration status. Valid values are: "ENABLE" or "DISABLE".
registrationConfig	RegistrationConfig	Information about the registration configuration.
templateBody	string	The template body.
roleArn	string length max:2048 min:20	The ARN of the role.
removeAutoRegistration	boolean	If true, remove auto registration.

Output:

None

UpdateCertificate

Updates the status of the specified certificate. This operation is idempotent.

Moving a certificate from the ACTIVE state (including REVOKED) will not disconnect currently connected devices, but these devices will be unable to reconnect.

The ACTIVE state is required to authenticate devices connecting to AWS IoT using a certificate.

Request syntax:

```
PUT /certificates/certificateId?newStatus=newStatus
```

URI Request Parameters:

Name	Type	Req?	Description
certificateId	CertificateId	yes	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)
newStatus	CertificateStatus	yes	The new status. **Note:** Setting the status to PENDING_TRANSFER will result in an exception being thrown. PENDING_TRANSFER is a status used internally by AWS IoT. It is not intended for developer use. **Note:** The status value REGISTER_INACTIVE is deprecated and should not be used.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`CertificateStateException`

The certificate operation is not allowed.

HTTP response code: 406

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  update-certificate \
    --certificate-id <value> \
    --new-status <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "certificateId": "string",
  "newStatus": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
certificateId	string length max:64 min:64 pattern: (0x)?[a-fA-F0-9]+	The ID of the certificate. (The last part of the certificate ARN contains the certificate ID.)
newStatus	string enum: ACTIVE | INACTIVE | REVOKED | PENDING_TRANSFER | REGISTER_INACTIVE | PENDING_ACTIVATION java class: iot.identity.service.CertificateStatus	The new status. **Note:** Setting the status to PENDING_TRANSFER will result in an exception being thrown. PENDING_TRANSFER is a status used internally by AWS IoT. It is not intended for developer use. **Note:** The status value REGISTER_INACTIVE is deprecated and should not be used.

Output:

None

UpdateEventConfigurations

Updates the event configurations.

Request syntax:

```
PATCH /event-configurations
Content-type: application/json

{
  "eventConfigurations": {
    "string": {
      "Enabled": "boolean"
    }
  }
}
```

Request Body Parameters:

Name	Type	Req?	Description
eventConfigurations	EventConfigurations	no	The new event configuration values.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

CLI

Synopsis:

```
aws iot  update-event-configurations \
    [--event-configurations <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "eventConfigurations": {
    "string": {
      "Enabled": "boolean"
    }
  }
```

```
}
```

`cli-input-json` fields:

Name	Type	Description
eventConfigurations	map key: EventType value: Configuration	The new event configuration values.
EventType	string enum: THING | THING_GROUP | THING_TYPE | THING_GROUP_MEMBERSHIP | THING_GROUP_HIERARCHY | THING_TYPE_ASSOCIATION | JOB | JOB_EXECUTION java class: com.amazonaws.iot.common.types.enums.EventType	
Configuration	Configuration	
Enabled	boolean	True to enable the configuration.

Output:

None

UpdateIndexingConfiguration

Updates the search configuration.

Request syntax:

```
POST /indexing/config
Content-type: application/json

{
  "thingIndexingConfiguration": {
    "thingIndexingMode": "string"
  }
}
```

Request Body Parameters:

Name	Type	Req?	Description
thingIndexingConfiguration	ThingIndexingConfiguration	no	Thing indexing configuration.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  update-indexing-configuration \
    [--thing-indexing-configuration <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingIndexingConfiguration": {
    "thingIndexingMode": "string"
  }
}
```

`cli-input-json` fields:

Name	Type	Description
thingIndexingConfiguration	ThingIndexingConfiguration	Thing indexing configuration.
thingIndexingMode	string enum: OFF | REGISTRY | REGISTRY_AND_SHADOW java class: com.amazonaws.iot.indexing.ThingIndexingMode	Thing indexing mode. Valid values are: • REGISTRY – Your thing index will contain only registry data. • REGISTRY_AND_SHADOW - Your thing index will contain registry and shadow data.

Name	Type	Description
		• OFF - Thing indexing is disabled.

Output:

None

UpdateJobExecution

Updates the status of a job execution.

Request syntax:

```
POST /things/thingName/jobs/jobId
Content-type: application/json

{
  "status": "string",
  "statusDetails": {
    "string": "string"
  },
  "expectedVersion": "long",
  "includeJobExecutionState": "boolean",
  "includeJobDocument": "boolean",
  "executionNumber": "long"
}
```

URI Request Parameters:

Name	Type	Req?	Description
jobId	JobId	yes	The unique identifier assigned to this job when it was created.
thingName	ThingName	yes	The name of the thing associated with the device.

Request Body Parameters:

Name	Type	Req?	Description
status	JobExecutionStatus	yes	The new status for the job execution (IN_PROGRESS, FAILED, SUCCESS, or REJECTED). This must be specified on every update.
statusDetails	DetailsMap	no	Optional. A collection of name/value pairs

Name	Type	Req?	Description
			that describe the status of the job execution. If not specified, the statusDetails are unchanged.
expectedVersion	ExpectedVersion	no	Optional. The expected current version of the job execution. Each time you update the job execution, its version is incremented. If the version of the job execution stored in Jobs does not match, the update is rejected with a VersionMismatch error, and an ErrorResponse that contains the current job execution status data is returned. (This makes it unnecessary to perform a separate DescribeJobExecution request in order to obtain the job execution status data.)
includeJobExecutionState	IncludeExecutionState	no	Optional. When included and set to true, the response contains the JobExecutionState data. The default is false.
includeJobDocument	IncludeJobDocument	no	Optional. When set to true, the response contains the job document. The default is false.
executionNumber	ExecutionNumber	no	Optional. A number that identifies a particular job execution on a particular device.

Response syntax:

```
Content-type: application/json

{
  "executionState": {
    "status": "string",
    "statusDetails": {
```

```
      "string": "string"
    },
    "versionNumber": "long"
  },
  "jobDocument": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
executionState	JobExecutionState	no	A JobExecutionState object.
jobDocument	JobDocument	no	The contents of the Job Documents.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`CertificateValidationException`

The certificate is invalid.

HTTP response code: 400

`InvalidStateTransitionException`

An update attempted to change the job execution to a state that is invalid because of the job execution's current state (for example, an attempt to change a request in state SUCCESS to state IN_PROGRESS). In this case, the body of the error message also contains the executionState field.

HTTP response code: 409

CLI

Synopsis:

```
aws iot  update-job-execution \
    --job-id <value> \
    --thing-name <value> \
    --status <value> \
    [--status-details <value>] \
    [--expected-version <value>] \
    [--include-job-execution-state | --no-include-job-execution-state] \
    [--include-job-document | --no-include-job-document] \
    [--execution-number <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "jobId": "string",
  "thingName": "string",
  "status": "string",
  "statusDetails": {
    "string": "string"
  },
  "expectedVersion": "long",
  "includeJobExecutionState": "boolean",
  "includeJobDocument": "boolean",
  "executionNumber": "long"
}
```

`cli-input-json` fields:

Name	Type	Description
jobId	string length max:64 min:1 pattern: [a-zA-Z0-9_-]+	The unique identifier assigned to this job when it was created.
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing associated with the device.
status	string enum: QUEUED | IN_PROGRESS | SUCCEEDED | FAILED | REJECTED | REMOVED | CANCELED java class: com.amazonaws.iot.laser.common.JobExecutionStatus	The new status for the job execution (IN_PROGRESS, FAILED, SUCCESS, or REJECTED). This must be specified on every update.
statusDetails	map key: DetailsKey value: DetailsValue	Optional. A collection of name/value pairs that describe the status of the job execution. If not specified, the statusDetails are unchanged.
DetailsKey	string length max:128 min:1	

Name	Type	Description
	pattern: [a-zA-Z0-9:_-]+	
DetailsValue	string length max:1024 min:1 pattern: [^\\p{C}]*+	
expectedVersion	long java class: java.lang.Long	Optional. The expected current version of the job execution. Each time you update the job execution, its version is incremented. If the version of the job execution stored in Jobs does not match, the update is rejected with a VersionMismatch error, and an ErrorResponse that contains the current job execution status data is returned. (This makes it unnecessary to perform a separate DescribeJobExecution request in order to obtain the job execution status data.)
includeJobExecutionState	boolean java class: java.lang.Boolean	Optional. When included and set to true, the response contains the JobExecutionState data. The default is false.
includeJobDocument	boolean java class: java.lang.Boolean	Optional. When set to true, the response contains the job document. The default is false.
executionNumber	long java class: java.lang.Long	Optional. A number that identifies a particular job execution on a particular device.

Output:

```
{
  "executionState": {
    "status": "string",
    "statusDetails": {
      "string": "string"
    },
    "versionNumber": "long"
  },
  "jobDocument": "string"
}
```

cli output fields:

Name	Type	Description
executionState	JobExecutionState	A JobExecutionState object.

Name	Type	Description
status	string enum: QUEUED | IN_PROGRESS | SUCCEEDED | FAILED | REJECTED | REMOVED | CANCELED java class: com.amazonaws.iot.laser.common.JobExecutionStatus	The status of the job execution. Can be one of: "QUEUED", "IN_PROGRESS", "FAILED", "SUCCESS", "CANCELED", "REJECTED", or "REMOVED".
statusDetails	map key: DetailsKey value: DetailsValue	A collection of name/value pairs that describe the status of the job execution.
DetailsKey	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
DetailsValue	string length max:1024 min:1 pattern: [^\\p{C}]*+	
versionNumber	long	The version of the job execution. Job execution versions are incremented each time they are updated by a device.
jobDocument	string length max:32768	The contents of the Job Documents.

UpdateRoleAlias

Updates a role alias.

Request syntax:

```
PUT /role-aliases/roleAlias
Content-type: application/json

{
  "roleArn": "string",
  "credentialDurationSeconds": "integer"
}
```

URI Request Parameters:

Name	Type	Req?	Description
roleAlias	RoleAlias	yes	The role alias to update.

Request Body Parameters:

Name	Type	Req?	Description
roleArn	RoleArn	no	The role ARN.
credentialDurationSeconds	CredentialDurationSeconds	no	The number of seconds the credential will be valid.

Response syntax:

```
Content-type: application/json

{
  "roleAlias": "string",
  "roleAliasArn": "string"
}
```

Response Body Parameters:

Name	Type	Req?	Description
roleAlias	RoleAlias	no	The role alias.
roleAliasArn	RoleAliasArn	no	The role alias ARN.

Errors:

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  update-role-alias \
    --role-alias <value> \
    [--role-arn <value>] \
    [--credential-duration-seconds <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "roleAlias": "string",
  "roleArn": "string",
  "credentialDurationSeconds": "integer"
}
```

`cli-input-json` fields:

Name	Type	Description
roleAlias	string length max:128 min:1 pattern: [w=,@-]+	The role alias to update.
roleArn	string length max:2048 min:20	The role ARN.
credentialDurationSeconds	integer java class: java.lang.Integer range- max:3600 min:900	The number of seconds the credential will be valid.

Output:

```
{
  "roleAlias": "string",
  "roleAliasArn": "string"
}
```

cli output fields:

Name	Type	Description
roleAlias	string	The role alias.

Name	Type	Description
	length max:128 min:1 pattern: [w=,@-]+	
roleAliasArn	string	The role alias ARN.

UpdateStream

Updates an existing stream. The stream version will be incremented by one.

Request syntax:

```
PUT /streams/streamId
Content-type: application/json

{
  "description": "string",
  "files": [
    {
      "fileId": "integer",
      "s3Location": {
        "bucket": "string",
        "key": "string",
        "version": "string"
      }
    }
  ],
  "roleArn": "string"
}
```

URI Request Parameters:

Name	Type	Req?	Description
streamId	StreamId	yes	The stream ID.

Request Body Parameters:

Name	Type	Req?	Description
description	StreamDescription	no	The description of the stream.
files	StreamFiles	no	The files associated with the stream.
roleArn	RoleArn	no	An IAM role that allows the IoT service principal assumes to access your S3 files.

Response syntax:

```
Content-type: application/json
```

```
{
  "streamId": "string",
  "streamArn": "string",
  "description": "string",
  "streamVersion": "integer"
}
```

Response Body Parameters:

Name	Type	Req?	Description
streamId	StreamId	no	The stream ID.
streamArn	StreamArn	no	The stream ARN.
description	StreamDescription	no	A description of the stream.
streamVersion	StreamVersion	no	The stream version.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

CLI

Synopsis:

```
aws iot  update-stream \
    --stream-id <value> \
    [--description <value>] \
    [--files <value>] \
    [--role-arn <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "streamId": "string",
  "description": "string",
  "files": [
    {
      "fileId": "integer",
      "s3Location": {
        "bucket": "string",
        "key": "string",
        "version": "string"
      }
    }
  ],
  "roleArn": "string"
}
```

`cli-input-json` fields:

Name	Type	Description
streamId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The stream ID.
description	string length max:2028 pattern: [^\\p{C}]+	The description of the stream.
files	list member: StreamFile	The files associated with the stream.
StreamFile	StreamFile	
fileId	integer java class: java.lang.Integer range- max:255 min:0	The file ID.
s3Location	S3Location	The location of the file in S3.
bucket	string length min:1	The S3 bucket that contains the file to stream.
key	string	The name of the file within the S3 bucket to stream.

Name	Type	Description
	length min:1	
version	string	The file version.
roleArn	string length max:2048 min:20	An IAM role that allows the IoT service principal assumes to access your S3 files.

Output:

```
{
  "streamId": "string",
  "streamArn": "string",
  "description": "string",
  "streamVersion": "integer"
}
```

cli output fields:

Name	Type	Description
streamId	string length max:128 min:1 pattern: [a-zA-Z0-9_-]+	The stream ID.
streamArn	string	The stream ARN.
description	string length max:2028 pattern: [^\\p{C}]+	A description of the stream.
streamVersion	integer java class: java.lang.Integer range- max:65535 min:0	The stream version.

UpdateThing

Updates the data for a thing.

Request syntax:

```
PATCH /things/thingName
Content-type: application/json

{
  "thingTypeName": "string",
  "attributePayload": {
    "attributes": {
      "string": "string"
    },
    "merge": "boolean"
```

```
  },
  "expectedVersion": "long",
  "removeThingType": "boolean"
}
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	yes	The name of the thing to update.

Request Body Parameters:

Name	Type	Req?	Description
thingTypeName	ThingTypeName	no	The name of the thing type.
attributePayload	AttributePayload	no	A list of thing attributes, a JSON string containing name-value pairs. For example: `\"attributes\":{\"name1\":\"value2\"}` This data is used to add new attributes or update existing attributes.
expectedVersion	OptionalVersion	no	The expected version of the thing record in the registry. If the version of the record in the registry does not match the expected version specified in the request, the `UpdateThing` request is rejected with a `VersionConflictException`.
removeThingType	RemoveThingType	no	Remove a thing type association. If **true**, the association is removed.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`VersionConflictException`

An exception thrown when the version of a thing passed to a command is different than the version specified with the --version parameter.

HTTP response code: 409

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  update-thing \
    --thing-name <value> \
    [--thing-type-name <value>] \
    [--attribute-payload <value>] \
    [--expected-version <value>] \
    [--remove-thing-type | --no-remove-thing-type]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string",
  "thingTypeName": "string",
  "attributePayload": {
    "attributes": {
      "string": "string"
    },
    "merge": "boolean"
  },
  "expectedVersion": "long",
```

```
  "removeThingType": "boolean"
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing to update.
thingTypeName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing type.
attributePayload	AttributePayload	A list of thing attributes, a JSON string containing name-value pairs. For example: `\"attributes\": {\"name1\":\"value2\"}` This data is used to add new attributes or update existing attributes.
attributes	map key: AttributeName value: AttributeValue	A JSON string containing up to three key-value pair in JSON format. For example: `\"attributes\": {\"string1\": \"string2\"}`
AttributeName	string length max:128 pattern: [a-zA-Z0-9_.,@/:#-]+	
AttributeValue	string length max:800 pattern: [a-zA-Z0-9_.,@/:#-]*	
merge	boolean	Specifies whether the list of attributes provided in the `AttributePayload` is merged with the attributes stored in the registry, instead of overwriting them. To remove an attribute, call `UpdateThing` with an empty attribute value.

Name	Type	Description
		Note The `merge` attribute is only valid when calling `UpdateThing`.
expectedVersion	long java class: java.lang.Long	The expected version of the thing record in the registry. If the version of the record in the registry does not match the expected version specified in the request, the `UpdateThing` request is rejected with a `VersionConflictException`.
removeThingType	boolean	Remove a thing type association. If **true**, the association is removed.

Output:

None

UpdateThingGroup

Update a thing group.

Request syntax:

```
PATCH /thing-groups/thingGroupName
Content-type: application/json

{
  "thingGroupProperties": {
    "thingGroupDescription": "string",
    "attributePayload": {
      "attributes": {
        "string": "string"
      },
      "merge": "boolean"
    }
  },
  "expectedVersion": "long"
}
```

URI Request Parameters:

Name	Type	Req?	Description
thingGroupName	ThingGroupName	yes	The thing group to update.

Request Body Parameters:

Name	Type	Req?	Description
thingGroupProperties	ThingGroupProperties	yes	The thing group properties.
expectedVersion	OptionalVersion	no	The expected version of the thing group. If this does not match the version of the thing group being updated, the update will fail.

Response syntax:

```
Content-type: application/json

{
  "version": "long"
}
```

Response Body Parameters:

Name	Type	Req?	Description
version	Version	no	The version of the updated thing group.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`VersionConflictException`

An exception thrown when the version of a thing passed to a command is different than the version specified with the --version parameter.

HTTP response code: 409

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  update-thing-group \
    --thing-group-name <value> \
    --thing-group-properties <value> \
    [--expected-version <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingGroupName": "string",
  "thingGroupProperties": {
    "thingGroupDescription": "string",
    "attributePayload": {
      "attributes": {
        "string": "string"
      },
      "merge": "boolean"
    }
  },
  "expectedVersion": "long"
}
```

`cli-input-json` fields:

Name	Type	Description
thingGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The thing group to update.
thingGroupProperties	ThingGroupProperties	The thing group properties.
thingGroupDescription	string length max:2028 pattern: [\\p{Graph}]*	The thing group description.
attributePayload	AttributePayload	The thing group attributes in JSON format.
attributes	map key: AttributeName value: AttributeValue	A JSON string containing up to three key-value pair in JSON format. For example: *\"attributes\": {\"string1\": \"string2\"}*
AttributeName	string	

Name	Type	Description
	length max:128 pattern: [a-zA-Z0-9_.,@/:#-]+	
AttributeValue	string length max:800 pattern: [a-zA-Z0-9_.,@/:#-]*	
merge	boolean	Specifies whether the list of attributes provided in the `AttributePayload` is merged with the attributes stored in the registry, instead of overwriting them. To remove an attribute, call `UpdateThing` with an empty attribute value. **Note** The `merge` attribute is only valid when calling `UpdateThing`.
expectedVersion	long java class: java.lang.Long	The expected version of the thing group. If this does not match the version of the thing group being updated, the update will fail.

Output:

```
{
  "version": "long"
}
```

cli output fields:

Name	Type	Description
version	long	The version of the updated thing group.

UpdateThingGroupsForThing

Updates the groups to which the thing belongs.

Request syntax:

```
PUT /thing-groups/updateThingGroupsForThing
Content-type: application/json
```

```
{
  "thingName": "string",
  "thingGroupsToAdd": [
    "string"
  ],
  "thingGroupsToRemove": [
    "string"
  ]
}
```

Request Body Parameters:

Name	Type	Req?	Description
thingName	ThingName	no	The thing whose group memberships will be updated.
thingGroupsToAdd	ThingGroupList	no	The groups to which the thing will be added.
thingGroupsToRemove	ThingGroupList	no	The groups from which the thing will be removed.

Errors:

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`ResourceNotFoundException`

The specified resource does not exist.

HTTP response code: 404

CLI

Synopsis:

```
aws iot  update-thing-groups-for-thing \
    [--thing-name <value>] \
    [--thing-groups-to-add <value>] \
```

```
    [--thing-groups-to-remove <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string",
  "thingGroupsToAdd": [
    "string"
  ],
  "thingGroupsToRemove": [
    "string"
  ]
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The thing whose group memberships will be updated.
thingGroupsToAdd	list member: ThingGroupName	The groups to which the thing will be added.
ThingGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	
thingGroupsToRemove	list member: ThingGroupName	The groups from which the thing will be removed.
ThingGroupName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	

Output:

None

UpdateThingShadow

Updates the thing shadow for the specified thing.

For more information, see UpdateThingShadow in the AWS IoT Developer Guide.

Request syntax:

```
POST /things/thingName/shadow
Content-type: application/json

{
  "payload": "blob"
}
```

URI Request Parameters:

Name	Type	Req?	Description
thingName	ThingName	yes	The name of the thing.

Request Body Parameters:

Name	Type	Req?	Description
payload	JsonDocument	yes	The state information, in JSON format.

Response syntax:

```
Content-type: application/json

{
  "payload": "blob"
}
```

Response Body Parameters:

Name	Type	Req?	Description
payload	JsonDocument	no	The state information, in JSON format.

Errors:

`ConflictException`

The specified version does not match the version of the document.

HTTP response code: 409

`RequestEntityTooLargeException`

The payload exceeds the maximum size allowed.

HTTP response code: 413

`InvalidRequestException`

The contents of the request were invalid. For example, this code is returned when an UpdateJobExecution request contains invalid status details. The message contains details about the error.

HTTP response code: 400

`ThrottlingException`

The rate exceeds the limit.

HTTP response code: 429

`UnauthorizedException`

You are not authorized to perform this operation.

HTTP response code: 401

`ServiceUnavailableException`

The service is temporarily unavailable.

HTTP response code: 503

`InternalFailureException`

An unexpected error has occurred.

HTTP response code: 500

`MethodNotAllowedException`

The specified combination of HTTP verb and URI is not supported.

HTTP response code: 405

`UnsupportedDocumentEncodingException`

The encoding is not supported.

HTTP response code: 415

CLI

Synopsis:

```
aws iot  update-thing-shadow \
    --thing-name <value> \
    --payload <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
  "thingName": "string",
  "payload": "blob"
}
```

`cli-input-json` fields:

Name	Type	Description
thingName	string length max:128 min:1 pattern: [a-zA-Z0-9:_-]+	The name of the thing.

Name	Type	Description
payload	blob	The state information, in JSON format.

Output:

```
{
  "payload": "blob"
}
```

cli output fields:

Name	Type	Description
payload	blob	The state information, in JSON format.