

---

# AWS IoT

## Developer Guide



## AWS IoT: Developer Guide

Copyright © 2020 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

## Table of Contents

What Is AWS IoT .....	1
AWS IoT Components .....	1
How to Get Started with AWS IoT .....	2
Accessing AWS IoT .....	3
Related Services .....	3
How AWS IoT Works .....	3
Getting Started with AWS IoT Core .....	5
Sign in to the AWS IoT Console .....	5
Create a Thing .....	6
Register a Device .....	6
Create and Activate a Device Certificate .....	6
Create an AWS IoT Core Policy .....	8
Attach an AWS IoT Core Policy to a Device Certificate .....	9
Attach a Certificate to a Thing .....	10
Configure Your Device .....	11
View Device MQTT Messages with the AWS IoT MQTT Client .....	12
Configure and Test Rules .....	13
Create an SNS Topic .....	14
Subscribe to an Amazon SNS Topic .....	16
Create a Rule .....	18
Test the Amazon SNS Rule .....	24
Next Steps .....	24
Create and Track an AWS IoT Core Job .....	24
Connect Your Device to AWS IoT .....	24
Run the Jobs Sample .....	25
Create a Job Document .....	25
Create a Job .....	25
Execute the Job on a Device .....	33
Tracking Job Progress with Job and Job Execution Events .....	34
AWS IoT Rules Tutorials .....	38
Creating a Rule with a DynamoDB Action .....	38
Testing a Rule with a DynamoDB Action .....	48
Creating a Rule with a AWS Lambda Action .....	49
Create a Lambda Function .....	49
Test Your Lambda Function .....	54
Create a Rule with a Lambda Action .....	56
Test Your Rule with a Lambda Action .....	66
Troubleshooting a Rule with a Lambda Action .....	67
Creating an Amazon SNS Rule .....	68
Using the AWS IoT Device SDKs on a Raspberry Pi .....	77
Prerequisites .....	77
Create an AWS IoT Thing for Your Raspberry Pi .....	77
Using the AWS IoT Device SDK for Embedded C .....	78
Install the AWS IoT Device SDK for Embedded C .....	78
Sample App Configuration .....	79
Run Sample Applications .....	80
Using the AWS IoT Device SDK for JavaScript and Node .....	80
Install the Latest Version of Node.js .....	80
Install the AWS IoT Device SDK for JavaScript .....	81
Prepare to Run the Sample Applications .....	81
Run the Sample Applications .....	82
Other AWS IoT Tutorials .....	84
Monitoring Soil Moisture with AWS IoT and Raspberry Pi .....	84
Prerequisites .....	84

Setting Up AWS IoT .....	84
Setting Up Your Raspberry Pi and Moisture Sensor .....	88
Managing Devices with AWS IoT .....	93
How to Manage Things with the Registry .....	93
Create a Thing .....	93
List Things .....	94
Search for Things .....	94
Update a Thing .....	95
Delete a Thing .....	96
Attach a Principal to a Thing .....	96
Detach a Principal from a Thing .....	96
Thing Types .....	96
Create a Thing Type .....	97
List Thing Types .....	97
Describe a Thing Type .....	97
Associate a Thing Type with a Thing .....	98
Deprecate a Thing Type .....	98
Delete a Thing Type .....	99
Static Thing Groups .....	99
Create a Static Thing Group .....	100
Describe a Thing Group .....	101
Add a Thing to a Static Thing Group .....	102
Remove a Thing from a Static Thing Group .....	102
List Things in a Thing Group .....	102
List Thing Groups .....	103
List Groups for a Thing .....	104
Update a Static Thing Group .....	105
Delete a Thing Group .....	105
Attach a Policy to a Static Thing Group .....	105
Detach a Policy from a Static Thing Group .....	106
List the Policies Attached to a Static Thing Group .....	106
List the Groups for a Policy .....	106
Get Effective Policies for a Thing .....	107
Test Authorization for MQTT Actions .....	107
Dynamic Thing Groups .....	108
Create a Dynamic Thing Group .....	109
Describe a Dynamic Thing Group .....	110
Update a Dynamic Thing Group .....	110
Delete a Dynamic Thing Group .....	111
Limitations and Conflicts .....	111
Tagging Your AWS IoT Resources .....	114
Tag Basics .....	114
Tag Restrictions and Limitations .....	115
Using Tags with IAM Policies .....	115
Billing Groups .....	116
Viewing Cost Allocation and Usage Data .....	117
Security .....	119
Security in AWS IoT .....	119
Authentication .....	120
AWS Training and Certification .....	120
X.509 Certificate Overview .....	120
Server Authentication .....	121
Client Authentication .....	123
Custom Authentication .....	129
Managing Device Certs .....	137
Server Authentication .....	137
Authorization .....	137

AWS Training and Certification .....	139
AWS IoT Core Policies .....	139
Authorizing Direct Calls to AWS Services .....	167
Cross Account Access .....	171
Data Protection .....	172
Transport Security in AWS IoT .....	173
Data Encryption .....	174
Identity and Access Management .....	174
Audience .....	175
Authenticating With IAM Identities .....	175
Managing Access Using Policies .....	177
How AWS IoT Works with IAM .....	178
Identity-Based Policy Examples .....	193
Troubleshooting .....	195
Logging and Monitoring .....	197
Monitoring Tools .....	197
Monitoring with Amazon CloudWatch .....	198
Monitoring with CloudWatch Logs .....	209
Logging AWS IoT API Calls with AWS CloudTrail .....	231
Compliance Validation .....	233
Resilience .....	234
Infrastructure Security .....	234
Vulnerability Analysis .....	234
Security Best Practices .....	235
Protecting MQTT Connections in AWS IoT .....	235
Keep Your Device's Clock in Sync .....	237
Validate the Server Certificate .....	237
Use a Single Identity Per Device .....	237
Use Just In Time Provisioning .....	238
AWS Training and Certification .....	238
Connecting Devices .....	239
Configurable Endpoints (Beta) .....	239
Creating and Configuring AWS-Managed Domains .....	240
Creating and Configuring Custom Domains .....	241
Managing Domain Configurations .....	243
Message Broker .....	244
Topics .....	244
Topic Names .....	244
Topic Filters .....	245
Reserved Topics .....	245
Protocols .....	253
Protocols, Port Mappings, and Authentication .....	253
MQTT .....	253
HTTP .....	258
Rules .....	260
Granting AWS IoT the Required Access .....	260
Pass Role Permissions .....	262
Creating an AWS IoT Rule .....	262
Viewing Your Rules .....	266
Deleting a Rule .....	266
AWS IoT Rule Actions .....	267
CloudWatch Alarm Action .....	267
CloudWatch Logs Action .....	268
CloudWatch Metric Action .....	269
DynamoDB Action .....	270
DynamoDBv2 Action .....	271
Elasticsearch Action .....	272

Firehose Action .....	273
HTTP Action .....	274
IoT Analytics Action .....	276
IoT Events Action .....	277
IoT SiteWise Action .....	278
Kinesis Action .....	281
Lambda Action .....	282
Republish Action .....	284
S3 Action .....	285
Salesforce Action .....	286
SNS Action .....	286
SQS Action .....	287
Step Functions Action .....	288
Troubleshooting a Rule .....	289
Error Handling (Error Action) .....	289
Error Action Message Format .....	289
Error Action Example .....	290
Working with Topic Rule Destinations .....	291
Creating a Topic Rule Destination .....	292
Confirming a Topic Rule Destination .....	292
Disabling a Topic Rule Destination .....	292
Enabling a Topic Rule Destination .....	293
Sending a New Confirmation Message .....	293
Deleting a Topic Rule Destination .....	293
Reducing Messaging Costs with Basic Ingest .....	293
Using Basic Ingest .....	293
AWS IoT SQL Reference .....	294
SELECT Clause .....	295
FROM Clause .....	297
WHERE Clause .....	298
Data Types .....	298
Operators .....	301
Functions .....	307
Literals .....	342
Case Statements .....	343
JSON Extensions .....	343
Substitution Templates .....	344
Nested Object Queries .....	345
SQL Versions .....	346
Device Shadow Service .....	348
Device Shadow Service Data Flow .....	348
Detecting a Thing Is Connected .....	354
Device Shadow Service Documents .....	355
Document Properties .....	356
Versioning of a Device Shadow .....	356
Client Token .....	357
Example Document .....	357
Empty Sections .....	357
Arrays .....	358
Using Shadows .....	359
Protocol Support .....	359
Updating a Shadow .....	359
Retrieving a Shadow Document .....	360
Deleting Data .....	363
Deleting a Shadow .....	363
Delta State .....	364
Observing State Changes .....	365

Message Order .....	366
Trim Shadow Messages .....	367
RESTful API .....	367
GetThingShadow .....	368
UpdateThingShadow .....	368
DeleteThingShadow .....	369
MQTT Pub/Sub Topics .....	370
/update .....	370
/update/accepted .....	371
/update/documents .....	372
/update/rejected .....	372
/update/delta .....	373
/get .....	374
/get/accepted .....	374
/get/rejected .....	375
/delete .....	375
/delete/accepted .....	376
/delete/rejected .....	376
Document Syntax .....	377
Request State Documents .....	377
Response State Documents .....	377
Error Response Documents .....	379
Error Messages .....	379
Jobs .....	381
Jobs Key Concepts .....	381
Managing Jobs .....	383
Creating and Managing Jobs (Console) .....	384
Creating and Managing Jobs (CLI) .....	385
Devices and Jobs .....	393
Programming Devices to Work with Jobs .....	395
Using the AWS IoT Jobs APIs .....	404
Job Management and Control API .....	405
Jobs Device MQTT and HTTPS APIs .....	460
Job Rollout and Abort Configuration .....	484
Using Job Rollout Rates .....	484
Using Job Rollout Abort Configurations .....	485
Job Limits .....	486
AWS IoT Secure Tunneling .....	487
Secure Tunneling Concepts .....	487
Secure Tunneling Tutorial .....	487
Prerequisites .....	488
Open a Tunnel .....	488
Start the Local Proxy .....	488
Start an SSH Session .....	489
Close the Tunnel .....	489
Secure Tunnel Lifecycle .....	489
Controlling Access to Tunnels .....	490
Tunnel Access Prerequisites .....	490
iot:OpenTunnel .....	490
iot:DescribeTunnel .....	491
iot>ListTunnels .....	491
iot>ListTagsForResource .....	492
iot:CloseTunnel .....	492
iot:TagResource .....	492
iot:UntagResource .....	493
Local Proxy .....	493
Local Proxy Security Best Practices .....	493

IoT Agent Snippet .....	494
Configuring a Remote Device .....	495
Device Provisioning .....	496
Provisioning Devices That Don't Have Device Certificates Using Fleet Provisioning .....	496
Provisioning by Claim .....	497
Provisioning by Trusted User .....	498
Fleet Provisioning APIs .....	498
Provisioning Devices That Have Device Certificates .....	501
Single Thing Provisioning .....	502
Just-in-Time Provisioning .....	502
Bulk Registration .....	505
Provisioning Templates .....	505
Parameters Section .....	506
Resources Section .....	506
Template Example for JITP and Bulk Registration .....	510
Fleet Provisioning .....	511
Fleet Indexing Service .....	514
Managing Thing Indexing .....	514
Enabling Thing Indexing .....	514
Describing a Thing Index .....	520
Querying a Thing Index .....	520
Restrictions and Limitations .....	522
Authorization .....	523
Managing Thing Group Indexing .....	524
Enabling Thing Group Indexing .....	524
Describing Group Indexes .....	525
Querying a Thing Group Index .....	525
Authorization .....	525
Querying for Aggregate Data .....	525
GetStatistics .....	525
GetCardinality .....	527
GetPercentiles .....	528
Authorization .....	530
Query Syntax .....	530
Example Thing Queries .....	531
Example Thing Group Queries .....	532
AWS IoT Device Defender .....	534
AWS Training and Certification .....	534
Audit .....	534
Issue Severity .....	534
Next Steps .....	535
Audit Checks .....	535
How to Perform Audits .....	559
Audit Commands .....	566
Manage Audit Settings .....	566
Schedule Audits .....	570
Run an On-Demand Audit .....	579
Manage Audit Instances .....	580
Check Audit Results .....	586
Mitigation Actions .....	592
How to Define and Manage Mitigation Actions .....	595
Apply Mitigation Actions .....	598
Permissions .....	603
Mitigation Action Commands .....	606
CreateMitigationAction .....	607
UpdateMitigationAction .....	611
ListMitigationActions .....	614

DescribeMitigationAction .....	616
DeleteMitigationAction .....	620
StartAuditMitigationActionsTask .....	621
CancelAuditMitigationActionsTask .....	624
ListAuditMitigationActionsExecutions .....	624
ListAuditMitigationActionsTasks .....	627
DescribeAuditMitigationActionsTask .....	630
Detect .....	635
Concepts .....	636
Behaviors .....	637
Metrics .....	638
Scoping Metrics in Security Profiles Using Dimensions .....	649
How to Use Dimensions in the Console .....	650
Monitoring the Behavior of Unregistered Devices .....	653
How to Use AWS IoT Device Defender Detect .....	654
Permissions .....	655
Sending Metrics from Devices .....	656
Detect Commands .....	661
AttachSecurityProfile .....	662
CreateDimension .....	663
CreateSecurityProfile .....	664
DeleteDimension .....	666
DeleteSecurityProfile .....	667
DescribeDimension .....	667
DescribeSecurityProfile .....	668
DetachSecurityProfile .....	669
ListActiveViolations .....	670
ListDimensions .....	671
ListSecurityProfiles .....	672
ListSecurityProfilesForTarget .....	673
ListTargetsForSecurityProfile .....	673
ListViolationEvents .....	674
UpdateDimension .....	675
UpdateSecurityProfile .....	676
ValidateSecurityProfileBehaviors .....	678
Device Agent Integration with AWS IoT Greengrass .....	679
Security Best Practices for Device Agents .....	681
Event Messages .....	683
Registry Events .....	684
Jobs Events .....	690
Lifecycle Events .....	693
Connect/Disconnect Events .....	693
Subscribe/Unsubscribe Events .....	696
Alexa Voice Service Integration for AWS IoT .....	698
Getting Started with Alexa Voice Service Integration for AWS IoT on an NXP Device .....	699
Preview AVS Integration for AWS IoT with a Preconfigured NXP Account .....	700
Use Your AWS and Alexa Voice Service Developer Accounts to Set Up AVS for AWS IoT .....	703
AWS IoT Device and Mobile SDKs .....	706
AWS Mobile SDK for Android .....	706
Arduino Yún SDK .....	706
AWS IoT Device SDK for Embedded C .....	706
AWS IoT C++ Device SDK .....	707
AWS Mobile SDK for iOS .....	707
AWS IoT Device SDK for Java .....	707
AWS IoT Device SDK for JavaScript .....	707
AWS IoT Device SDK for Python .....	708
Troubleshooting .....	709

Diagnosing Connectivity Issues .....	709
Authentication .....	709
Authorization .....	709
Diagnosing Rules Issues .....	709
Diagnosing Problems with Shadows .....	710
Diagnosing Salesforce Action Issues .....	711
Execution Trace .....	711
Action Success and Failure .....	712
Troubleshooting Aggregation Queries for the Fleet Indexing Service .....	712
AWS IoT Device Defender Troubleshooting Guide .....	713
AWS IoT Errors .....	716
AWS IoT Quotas .....	717

# What Is AWS IoT?

AWS IoT provides secure, bi-directional communication between Internet-connected devices such as sensors, actuators, embedded micro-controllers, or smart appliances and the AWS Cloud. This enables you to collect telemetry data from multiple devices, 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:

### **Alexa Voice Service (AVS) Integration for AWS IoT**

Brings Alexa Voice to any connected device. AVS for AWS IoT reduces the cost and complexity of integrating Alexa. This feature leverages AWS IoT to offload intensive computational and memory audio tasks from the device to the cloud. Because of the resulting reduction in the engineering bill of materials (eBoM) cost, device makers can cost-effectively bring Alexa to resource-constrained IoT devices and enable consumers to talk directly to Alexa in parts of their home, office, or hotel rooms for an ambient experience.

AVS for AWS IoT enables Alexa built-in functionality on MCUs, such as the ARM Cortex M class with less than 1 MB embedded RAM. To do so, AVS offloads memory and compute tasks to a virtual Alexa Built-in device in the cloud. This reduces eBoM cost by up to 50 percent. For more information, see [Alexa Voice Service Integration for AWS IoT \(p. 698\)](#).

### **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, JSON Web Token verification, OAuth provider callout, and so on) and must return policy documents that are used by the device gateway to authorize MQTT operations. For more information, see [Custom Authentication \(p. 129\)](#).

### **Device gateway**

Enables devices to securely and efficiently communicate with AWS IoT.

### **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 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. For more information, see [Device Provisioning \(p. 496\)](#).

### **Device shadow**

A JSON document used to store and retrieve current state information for a device.

### Device Shadow service

Provides persistent representations of your devices in the AWS Cloud. You can publish updated state information to a device's shadow, and your device can synchronize its state when it connects. Your devices can also publish their current state to a shadow for use by applications or other devices. For more information, see [Device Shadow Service for AWS IoT \(p. 348\)](#).

### Group registry

Groups allow you to manage several devices 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 devices in it and in all of its child groups as well. Permissions given to a group will apply to all devices in the group and in all of its child groups. For more information, see [Managing Devices with AWS IoT \(p. 93\)](#).

### 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 devices 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 devices, groups or both. For more information, see [Jobs \(p. 381\)](#).

### Message broker

Provides a secure mechanism for devices 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. For more information, see [Message Broker for AWS IoT \(p. 244\)](#).

### Registry

Organizes the resources associated with each device in the AWS Cloud. You register your devices and associate up to three custom attributes with each one. You can also associate certificates and MQTT client IDs with each device to improve your ability to manage and troubleshoot them. For more information, see [Managing Devices with AWS IoT \(p. 93\)](#).

### 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. For more information, see [Rules for AWS IoT \(p. 260\)](#).

### Security and Identity service

Provides shared responsibility for security in the AWS Cloud. Your devices 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. For more information, see [Authentication \(p. 120\)](#).

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 device to AWS IoT, see [Getting Started with AWS IoT Core \(p. 5\)](#).

## Accessing AWS IoT

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

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

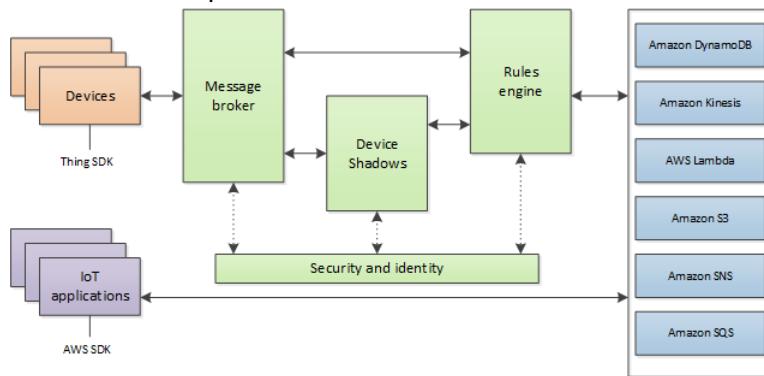
The state of each device connected to AWS IoT is stored in a device shadow. The Device Shadow service manages device shadows by responding to requests to retrieve or update device state data. The Device Shadow service makes it possible for devices to communicate with applications and for applications to communicate with devices.

Communication between a 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. In either case, the certificate

must be registered and activated with AWS IoT, and then copied onto your device. When your device communicates with AWS IoT, it presents the certificate to AWS IoT as a credential.

We recommend that all devices that connect to AWS IoT have an entry in the registry. The registry stores information about a device and the certificates that are used by the device 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 triggers 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.



# Getting Started with AWS IoT Core

This tutorial shows you how to create resources required to send, receive, and process MQTT messages from devices using AWS IoT Core. You use an MQTT client to emulate an IoT device.

For more information about AWS IoT Core, see [What Is AWS IoT Core \(p. 1\)?](#)

## Topics

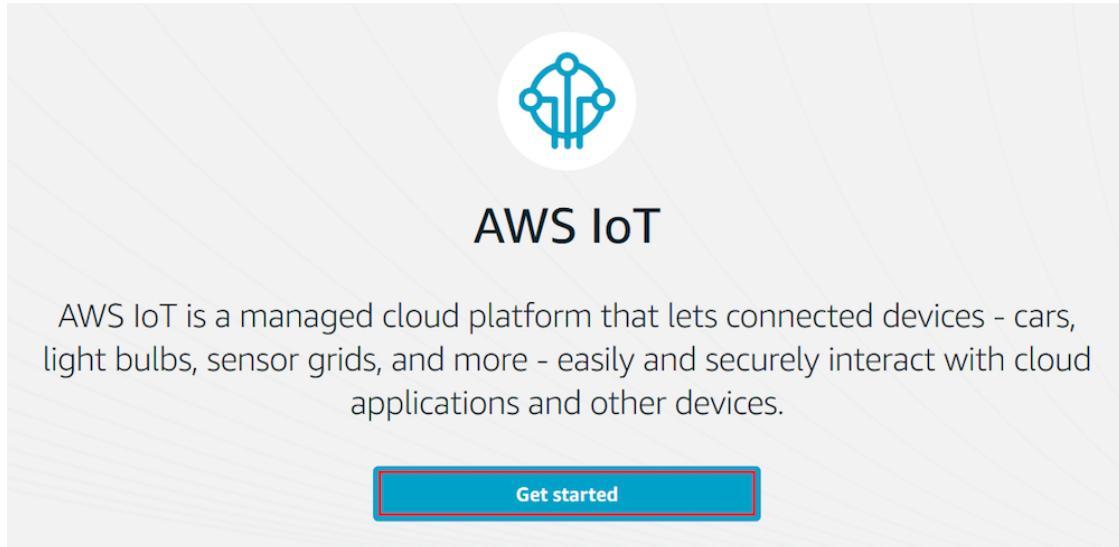
- [Sign in to the AWS IoT Console \(p. 5\)](#)
- [Create a Thing \(p. 6\)](#)
- [Register a Device \(p. 6\)](#)
- [Configure Your Device \(p. 11\)](#)
- [View Device MQTT Messages with the AWS IoT MQTT Client \(p. 12\)](#)
- [Configure and Test Rules \(p. 13\)](#)
- [Create and Track an AWS IoT Core Job \(p. 24\)](#)

## 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.

# Create a Thing

Devices connected to AWS IoT are represented by *things* in the AWS IoT registry. A *thing* represents a specific device or logical entity. It can be a physical device or sensor (for example, a light bulb or a switch on the 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).

## To create a thing

1. On the **Welcome to the AWS IoT Console** page, in the navigation pane, choose **Manage**.
2. On the **You don't have any things yet** page, choose **Register a thing**.
3. On the **Creating AWS IoT things** page, choose **Create a single thing**.
4. On the **Create a thing** page, in the **Name** field, enter a name for your thing, such as **MyIoTThing**. Choose **Next**. To change a thing's name, you must create a new thing, give it the new name, and then delete the old thing.

### Note

We do not recommend the use of personally identifiable information in your thing name.

# Register a Device

The registry allows you to keep a record of all of the devices that are registered to your AWS IoT Core account. The process of registering your device includes these steps:

### Topics

- [Create and Activate a Device Certificate \(p. 6\)](#)
- [Create an AWS IoT Core Policy \(p. 8\)](#)
- [Attach an AWS IoT Core Policy to a Device Certificate \(p. 9\)](#)
- [Attach a Certificate to a Thing \(p. 10\)](#)

## Create and Activate a Device Certificate

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

1. Choose **Create certificate**.

The screenshot shows the 'CREATE A THING' section of the AWS IoT console. The main heading is 'Add a certificate for your thing'. Below it, a note says 'A certificate is used to authenticate your device's connection to AWS IoT.' There are three options: 'One-click certificate creation (recommended)', 'Create with CSR', and 'Use my certificate'. The 'One-click certificate creation' option is described as generating a certificate, public key, and private key using AWS IoT's certificate authority. The 'Create with CSR' and 'Use my certificate' options are described as registering your CA certificate and using your own certificates for one or many devices. The 'Skip certificate and create thing' option is described as requiring a certificate to be added later.

**CREATE A THING**

## Add a certificate for your thing

A certificate is used to authenticate your device's connection to AWS IoT.

**One-click certificate creation (recommended)**

This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

**Create with CSR**

Upload your own certificate signing request (CSR) based on a private key you own.

**Use my certificate**

Register your CA certificate and use your own certificates for one or many devices.

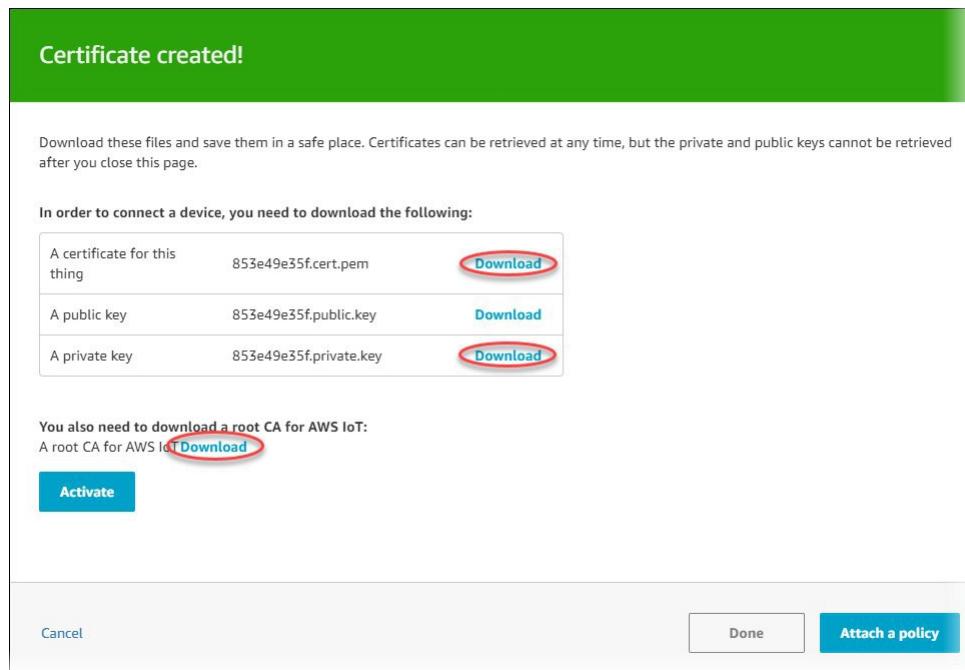
**Skip certificate and create thing**

You will need to add a certificate to your thing later before your device can connect to AWS IoT.

- On the **Certificate created** page, choose the **Download** links to download the certificate, private key, and root CA for AWS IoT Core. (You do not need to download the public key). Save each of them to your computer, and then choose **Activate** to continue.

**Note**

The root CA for AWS IoT **Download** link takes you to the [Server Authentication](#) page where you choose a CA certificate. Unlike the other **Download** links on the page, it doesn't directly download a file.



Be aware that the downloaded file names might be different from those listed on the **Certificate created** page. For example:

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

**Note**

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

3. Choose **Done** to return to the main page of the AWS IoT console.

When working with a device, you must copy the private key and root CA certificate onto your device. The instructions in this guide are written with the assumption that you are not using a device and are simply getting familiar with the AWS IoT console.

## Create an AWS IoT Core Policy

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

### To create an AWS IoT Core policy

1. In the left navigation pane, choose **Secure**, and then choose **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, enter a name for the policy (for example, **MyIotPolicy**). Do not use personally identifiable information in your policy names.

Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the [AWS IoT Policies documentation page](#).

Name

My\_IoT\_Policy

Add statements

Policy statements define the types of actions that can be performed by a resource.

Action

iot:\*

Resource ARN

\*

Effect

Allow  Deny

Remove

Add statement

Create

3. In the **Action** field, enter **iot:Connect**. In the **Resource ARN** field, enter \*. Select the **Allow** check box. This allows all clients to connect to AWS IoT Core.

**Note**

You can restrict which clients (devices) can 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>`

4. Choose the **Add Statement** button to add another policy statement. In the **Action** field, enter **iot:Publish**. In the **Resource ARN** field, enter the ARN of the topic to which your device publishes.

**Note**

The topic ARN follows this format:

`arn:aws:iot:<your-region>:<your-aws-account>:topic/<your/topic>`

For example:

`arn:aws:iot:us-east-1:123456789012:topic/my/topic`

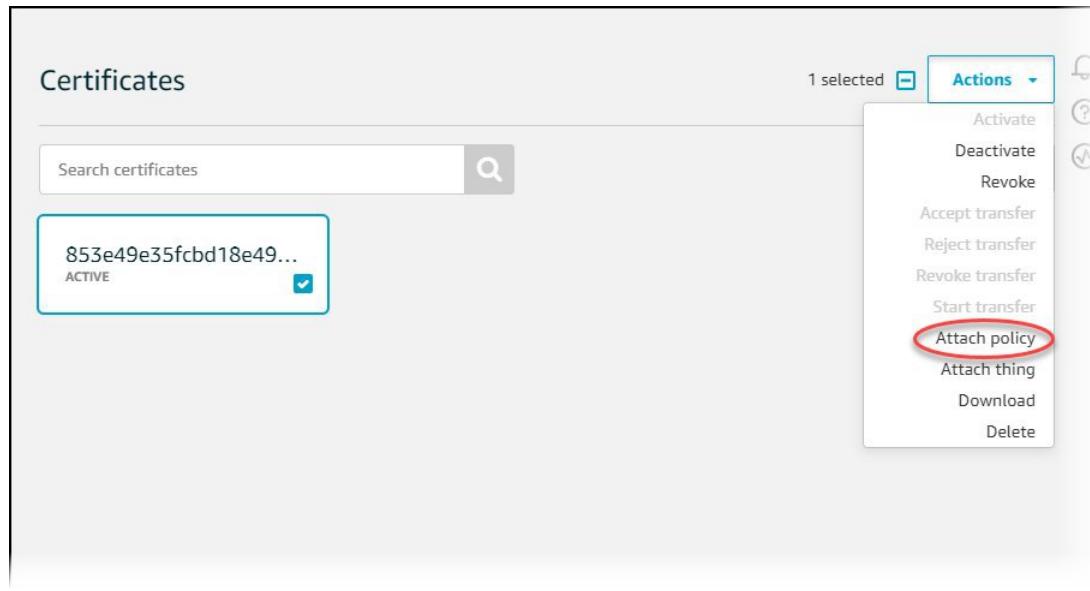
5. Finally, select the **Allow** check box. This allows your device to publish messages to the specified topic.
6. After you have entered the information for your policy, choose **Create**.

For more information, see [Managing AWS IoT Core Policies](#).

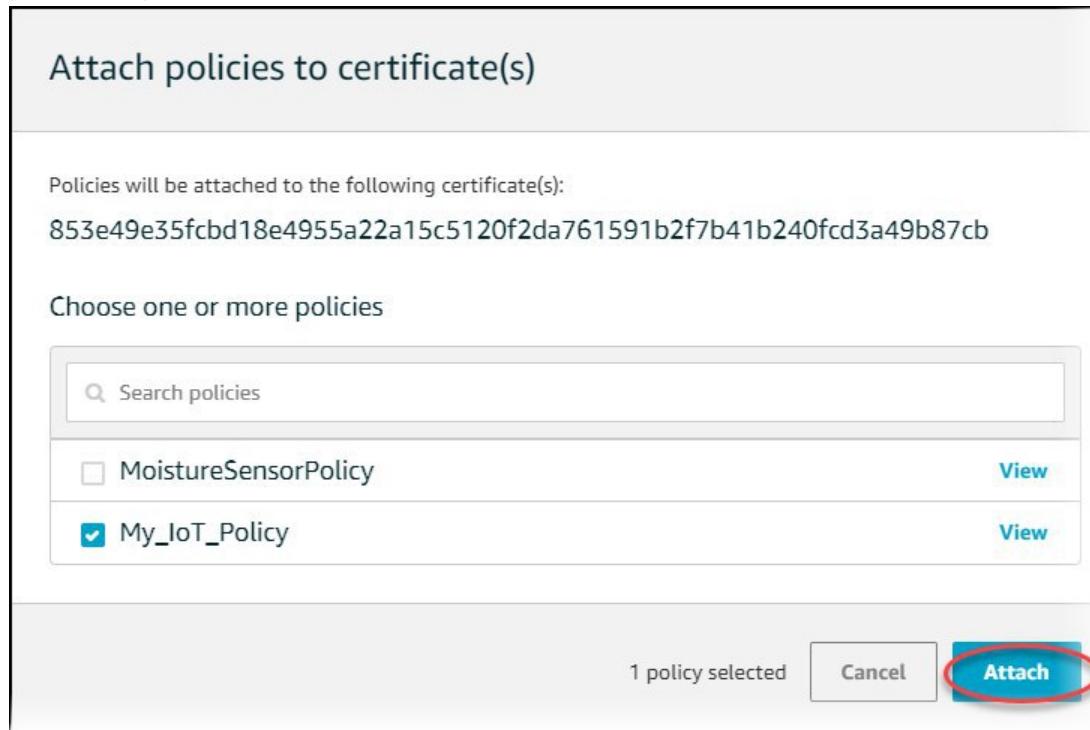
## Attach an AWS IoT Core Policy to a Device Certificate

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

1. In the left navigation pane, choose **Secure**, and then choose **Certificates**.
2. In the box for the certificate you created, choose ... to open a drop-down menu, and then choose **Attach policy**.



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



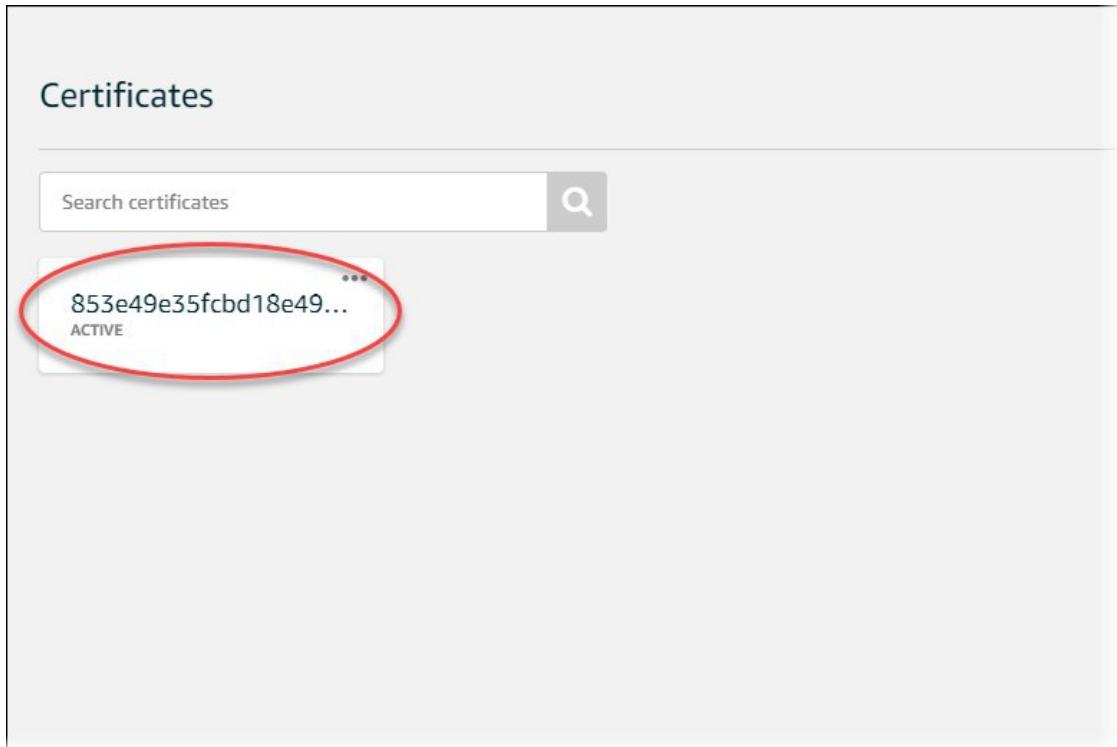
## Attach a Certificate to a Thing

A device must have a certificate, private key, and root CA certificate to authenticate with AWS IoT Core. We recommend that you also attach the device certificate to the IoT thing that represents your

device in AWS IoT Core. This allows you to create AWS IoT Core policies that grant permissions based on certificates attached to your things. For more information, see [Thing Policy Variables \(p. 143\)](#).

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

1. In the box for the certificate you created, choose ... to open a drop-down menu, and then choose **Attach thing**.
2. In **Attach things to certificate(s)**, select the check box next to the thing you registered, and then choose **Attach**.
3. To verify the thing is attached, select the box for 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

To communicate with AWS IoT Core, all devices must have a device certificate, private key, and root CA certificate installed. Consult your device's documentation to connect to it and copy your device certificate, private key, and root CA certificate onto your device.

If you don't have an IoT-ready device, you can use the MQTT client, the AWS IoT Device SDKs, or the AWS CLI. For more information, see the [Using the AWS IoT Device SDKs on a Raspberry Pi \(p. 77\)](#) section. The tutorials use a Raspberry Pi, but can easily be adapted for use with other types of computers.

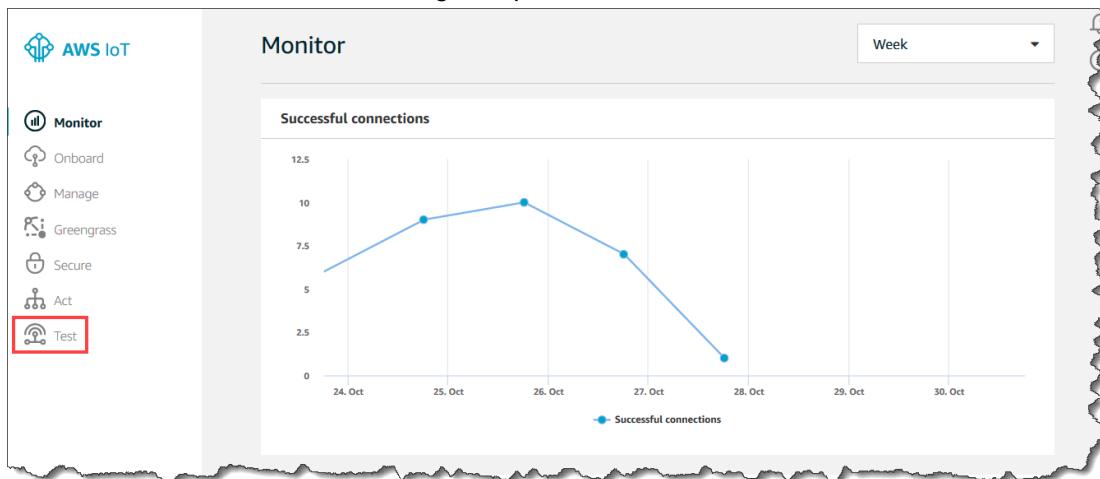
# View Device MQTT Messages with the AWS IoT MQTT Client

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

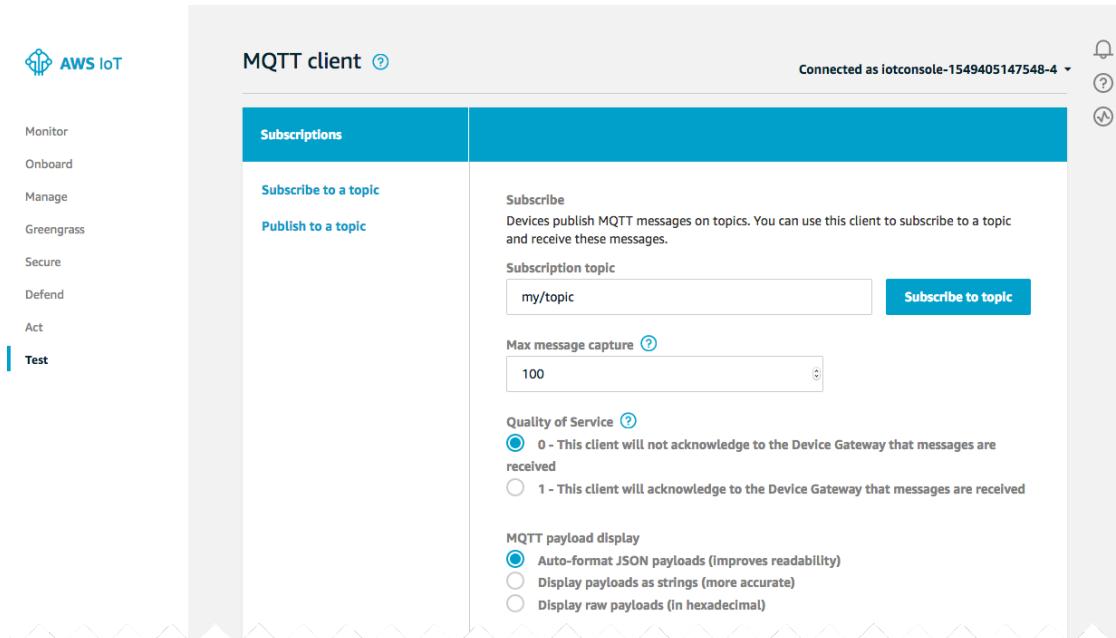
Devices publish MQTT messages on topics. You can use the AWS IoT MQTT client to subscribe to these topics to see these messages.

## 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 IoT thing publishes. Continuing with this example, in **Subscribe to a topic**, in the **Subscription topic** field, enter **my/topic**, and then choose **Subscribe to topic**.



The **my/topic** topic appears in the **Subscriptions** column.

The screenshot shows the AWS IoT MQTT client interface. On the left, there's a sidebar with options: Monitor, Onboard, Manage, Greengrass, Secure, Defend, Act, and Test. The 'Test' option is selected. The main area is titled 'MQTT client' and shows a subscription table with one entry: 'my/topic'. Below the table, there's a 'Publish' section with a text input field containing 'my/topic' and a 'Publish to topic' button. A code editor window shows the JSON message being published:

```

1 {
2   "message": "Hello, world",
3   "clientType": "MQTT client"
4 }

```

### To emulate an IoT thing sending a message

- On the MQTT client page, in the **Publish** section, in the **Specify a topic and a message to publish** field, enter **my/topic**. Do not use personally identifiable information in topic names.

In the message payload section, enter the following JSON:

```
{
  "message": "Hello, world",
  "clientType": "MQTT client"
}
```

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

The screenshot shows the 'Publish' section of the AWS IoT MQTT client. It includes a title 'Publish', a subtitle 'Specify a topic and a message to publish with a QoS of 0.', a text input field containing 'my/topic', and a 'Publish to topic' button. The 'Publish to topic' button is highlighted with a red rectangle. Below the input field, a code editor window shows the JSON message:

```

1 {
2   "message": "Hello, world",
3   "clientType": "MQTT client"
4 }

```

## 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 create and configure a rule to send the data received from a device to an Amazon SNS topic. Specifically, you:

- Create an Amazon SNS topic.
- Subscribe to the Amazon SNS topic using a cell phone number.
- Create a rule that sends a message to the Amazon SNS topic when a message is received from your device.
- Test the rule using the 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**.

The screenshot shows the Amazon SNS Dashboard. On the left sidebar, under the 'Topics' section, there is a red box around the 'Topics' link. The main dashboard area has a title 'Dashboard' and a subtitle 'Resources for us-west-2'. It displays two sections: 'Topics' (5) and 'Subscriptions' (2). Below these sections is a diagram titled 'Overview of Amazon SNS' illustrating the system-to-system messaging flow between a Publisher, Amazon SNS, and Subscribers (AWS Lambda, Amazon SQS, HTTPS).

3. Choose **Create topic**.

The screenshot shows the 'Topics' page in the Amazon SNS console. On the left sidebar, under the 'Topics' section, there is a red box around the 'Topics' link. The main page lists a single topic named 'MyTopic' with its ARN: arn:aws:sns:us-west-2:504350838278:MyTopic. There are buttons for 'Edit', 'Delete', 'Publish message', and a prominent orange 'Create topic' button.

4. Enter a topic name and display name, and then choose **Create topic**. Do not use personally identifiable information in Amazon SNS topic names.

Amazon SNS > Topics > Create topic

## Create topic

**Details**

Name  
 Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (\_).

Display name - *optional*  
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message. [Info](#)  
 Maximum 100 characters, including hyphens (-) and underscores (\_).

► **Encryption - optional**  
Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

► **Access policy - optional**  
This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic. [Info](#)

► **Delivery retry policy (HTTP/S) - optional**  
The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section. [Info](#)

► **Delivery status logging - optional**  
These settings configure the logging of message delivery status to CloudWatch Logs. [Info](#)

► **Tags - optional**  
A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. [Learn more](#)

[Cancel](#) [Create topic](#)

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

The screenshot shows the AWS IoT SNS Topic creation page. At the top, a green banner displays the message "Topic My\_IoT\_SNS\_Topic created successfully. You can create subscriptions and send messages to them from this topic." Below the banner, the topic name "My\_IoT\_SNS\_Topic" is displayed along with "Edit", "Delete", and "Publish message" buttons. The "Details" section shows the topic's Name ("My\_IoT\_SNS\_Topic"), Display name ("My IoT SNS Topic"), ARN ("arn:aws:sns:us-west-2:504350838278:My\_IoT\_SNS\_Topic"), and Topic owner ("504350838278"). Below the Details section are tabs for "Subscriptions", "Access policy", "Delivery retry policy (HTTP/S)", "Delivery status logging", "Encryption", and "Tags". The "Subscriptions" tab is selected, showing a table with columns: ID, Endpoint, Status, and Protocol. A message "No subscriptions found" is displayed, along with a "Create subscription" button.

## Subscribe to an Amazon SNS Topic

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

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

The screenshot shows the AWS IoT SNS Topics list page. On the left, a sidebar includes links for "SNS dashboard", "Topics" (which is selected), "Applications", "Subscriptions", and "Text messaging (SMS)". The main area displays a table of topics with columns: Name and ARN. A topic named "MyIoTButtonSNSTopic" is selected, indicated by a checked checkbox. A context menu is open over this topic, listing options: "Edit topic display name", "Subscribe to topic" (which is highlighted with a red border), "Confirm a subscription", "Edit topic policy", "Edit topic delivery policy", "Delivery status", and "Delete topics".

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

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

**Note**

Enter the phone number using numbers and dashes only.

Amazon SNS > Subscriptions > Create subscription

### Create subscription

**Details**

Topic ARN

Protocol  
The type of endpoint to subscribe

Endpoint  
A mobile number that can receive notifications from Amazon SNS.

ⓘ After your subscription is created, you must confirm it. [Info](#)

**► Subscription filter policy - optional**  
This policy filters the messages that a subscriber receives. [Info](#)

[Cancel](#) [Create subscription](#)

The Amazon SNS console displays the following message, but you might not receive a confirmation SMS message.

ⓘ Subscription to My\_IoT\_SNS\_Topic created successfully. X

The ARN of the subscription is arn:aws:sns:us-west-2:504350838278:My\_IoT\_SNS\_Topic:378721b7-6184-4908-97e5-998cced4afaf.

Amazon SNS > Topics > My\_IoT\_SNS\_Topic > Subscription: 378721b7-6184-4908-97e5-998cced4afaf

### Subscription: 378721b7-6184-4908-97e5-998cced4afaf

[Edit](#) [Delete](#)

**Details**

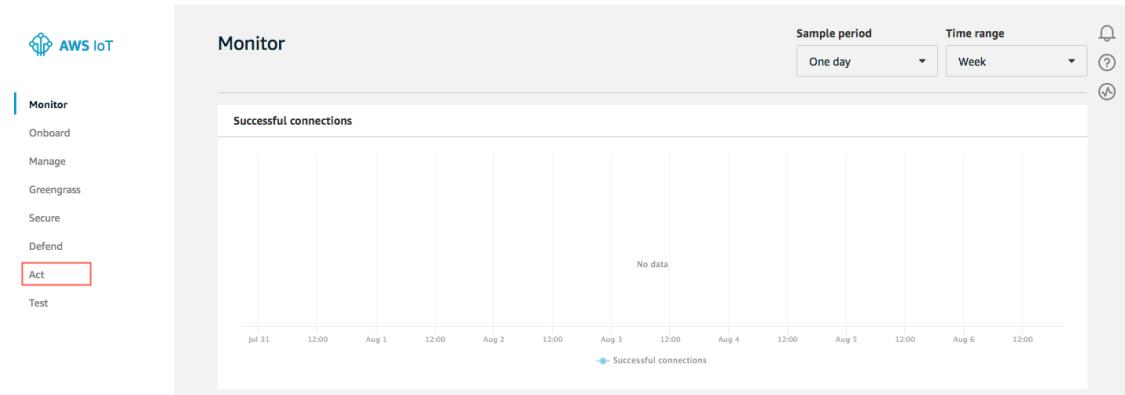
ARN arn:aws:sns:us-west-2:504350838278:My_IoT_SNS_Topic:378721b7-6184-4908-97e5-998cced4afaf	Status <span style="color: green;">Confirmed</span>
Endpoint +12065550100	Protocol SMS
Topic <a href="#">My_IoT_SNS_Topic</a>	

## Create a Rule

AWS IoT Core rules consist of a topic filter, rule action, and, in most cases, 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 Core rules engine continuously processes messages published on topics that match the topic filters defined in the rules.

In this example, you 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**.



### You don't have any rules yet

Rules give your things the ability to interact with AWS and other web services. Rules are analyzed and actions are performed based on the messages sent by your things.

[Learn more](#)

[Create a rule](#)

3. On the **Create a rule** page, in the **Name** field, enter a name for your rule.

#### Note

We do not recommend using personally identifiable information in your rule name.

In the **Description** field, enter a description for the rule.

## Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name  
MyIoTRule

Description  
A rule for AWS IoT Getting Started

4. Scroll down to **Rule query statement**. Choose the latest version from the **Using SQL version** drop-down list. In the **Rule query statement** field, enter **SELECT \* FROM 'my/topic'**.

**SELECT \*** specifies that you want to send the entire MQTT message that triggered the rule. **FROM 'my/topic'** is the topic filter. The rules engine uses the topic filter to determine which rules to trigger when an MQTT message is received.

### Rule query statement

Indicate the source of the messages you want to process with this rule.

#### Using SQL version

2016-03-23 ▾

### Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

1 SELECT \* FROM 'my/topic'

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

### Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (\*.required)

Add action

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

## Select an action

Select an action.

-  Insert a message into a DynamoDB table  
DYNAMODB
-  Split message into multiple columns of a DynamoDB table (DynamoDBv2)  
DYNAMODBV2
-  Send a message to a Lambda function  
LAMBDA
-  Send a message as an SNS push notification  
SNS
-  Send a message to an SQS queue  
SQS
-  Send a message to an Amazon Kinesis Stream  
AMAZON KINESIS
-  Republish a message to an AWS IoT topic  
AWS IOT REPUBLISH
-  Store a message in an Amazon S3 bucket  
S3
-  Send a message to an Amazon Kinesis Firehose stream  
AMAZON KINESIS FIREHOSE
-  Send message data to CloudWatch  
CLOUDWATCH METRICS
-  Change the state of a CloudWatch alarm  
CLOUDWATCH ALARMS
-  Send a message to the Amazon Elasticsearch Service  
AMAZON ELASTICSEARCH
-  Send a message to a Salesforce IoT Input Stream  
SALESFORCE IOT
-  Send a message to IoT Analytics  
IOT ANALYTICS
-  Send a message to an IoT Events Input  
IOT EVENTS
-  Start a Step Functions state machine execution  
STEP FUNCTIONS

Cancel

Configure action

7. On the **Configure action** page, under **SNS target**, choose **Select** to expand the SNS topic. Then choose **Select** next to the Amazon SNS topic you created earlier. Under **Message format**, choose **JSON**.

### Configure action

 Send a message as an SNS push notification  
SNS

\*SNS target

MyTopic	Create	Refresh	Clear	Close
Search				
MyTopic	Select			
My_IoT_SNS_Topic	Select			

Message format

JSON ▾

8. Now give AWS IoT Core permission to publish to the Amazon SNS topic on your behalf when the rule is triggered. Choose **Create a new role**. In **IAM role name**, enter a name for your new role, and then choose **Create a new role**.

### Create a new role

A new IAM role will be created in your account. An inline policy will be attached to the role providing scoped-down permissions allowing AWS IoT to access resources on your behalf.

Name

MySNSRole

Cancel Create role

9. Under **IAM role name**, choose **Update role** to apply the permissions to the newly created role. Choose the role, and then choose **Add action**.

### Configure action

 Send a message as an SNS push notification  
SNS

\*SNS target

My_IoT_SNS_Topic	<a href="#">Create</a>	<a href="#">Clear</a>	<a href="#">Select</a>
------------------	------------------------	-----------------------	------------------------

Message format

JSON	▼
------	---

Choose or create a role to grant AWS IoT access to perform this action.

MySNSRole	Policy Attached ✓	<a href="#">Create Role</a>	<a href="#">Select</a>
-----------	-------------------	-----------------------------	------------------------

[Cancel](#) Add action

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

## Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name

MyIoTRule

Description

A rule for AWS IoT Getting Started

### Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version

2016-03-23

Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

```
1 SELECT * FROM 'mytopic'
```

### Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (\*.required)



Send a message as an SNS push notification

My\_IoT\_SNS\_Topic

Remove Edit ▾

Add action

### Error action

Optionally set an action that will be executed when something goes wrong with processing your rule.

Add action

### Tags

Apply tags to your resources to help organize and identify them. A tag consists of a case-sensitive key-value pair. [Learn more](#) about tagging your AWS resources.

Tag name

Provide a tag name, e.g. Manufacturer

Value

Provide a tag value, e.g. Acme-Corporation

Clear

Add another

For more information about creating rules, see [AWS IoT Core Rules](#).

## Test the Amazon SNS Rule

You can use the AWS IoT MQTT client to test your rule.

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

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

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 Core rules, see [AWS IoT Core Rule Tutorials \(p. 38\)](#) and [AWS IoT Core Rules \(p. 260\)](#).

## Create and Track an AWS IoT Core Job

You can use AWS IoT Core jobs to deploy and track management tasks in your device fleet. You can use jobs to send remote actions to one or many devices at once, control the deployment of jobs to your devices, and track the current and past status of job executions for each device.

This topic shows you how to create and deploy a sample job to a device. It walks you through the steps required to create a job and track its events on a device that is configured to communicate with AWS IoT Core. These instructions are written with the assumption that you're using a Raspberry Pi, but they can be adapted for other Linux-based devices.

Here are some possible scenarios for using jobs:

- Updating device firmware, software, or files, such as security certificates.
- Performing administrative tasks, such as restarting devices or performing diagnostics.
- Restoring devices to factory settings or other known good states.

## Connect Your Device to AWS IoT

Perform the following steps to connect a Raspberry Pi to AWS IoT Core.

1. Follow the instructions in [Connecting Your Raspberry Pi](#). When you're finished, you'll have an AWS IoT Core thing registered in your AWS account. You'll also have fully configured security certificates on your device.
2. Complete the steps in the [Using the AWS IoT Device SDK for JavaScript](#) tutorial. When you're done, your device is connected to AWS IoT Core, and you can run the sample code that comes with the AWS IoT Device SDK for JavaScript.

Now your device is ready to use AWS IoT Core jobs.

## Run the Jobs Sample

The AWS IoT Device SDK for JavaScript includes a sample named [jobs-example.js](#). This sample can receive messages from the [AWS IoT console](#) to verify connectivity. It can also receive and process job executions that originate from the AWS IoT Core Jobs service.

You can run this sample by using the following command. Use the REST endpoint of your Raspberry Pi as the value of the `-H` parameter.

```
node examples/jobs-example.js -f ~/certs -H <PREFIX>.iot.<REGION>.amazonaws.com -T thingName
```

If you've created a configuration file that contains the thing name and the host endpoint (the REST endpoint of your device), you can use the following command.

```
node examples/jobs-example.js -f ./certs -F your config file name.json
```

## Create a Job Document

A job document is a JSON document that provides all of the information that your device needs to execute a job. The AWS IoT Device SDK for JavaScript uses a property named `operation` to route job documents to specific handlers. The `jobs-example.js` program has a sample handler for an operation named `customJob`. To create a job document named `example-job.json` for this handler, the file should contain the following JSON object.

```
{  
  "operation": "customJob",  
  "otherInfo": "someValue"  
}
```

For more sample job documents, see the documentation for the [jobs-agent.js](#) sample.

## Create a Job

Now you're ready to create a job that delivers the job document to all of the devices that you specify. You can use the AWS IoT console, the AWS IoT SDK, or the AWS IoT CLI to create a job.

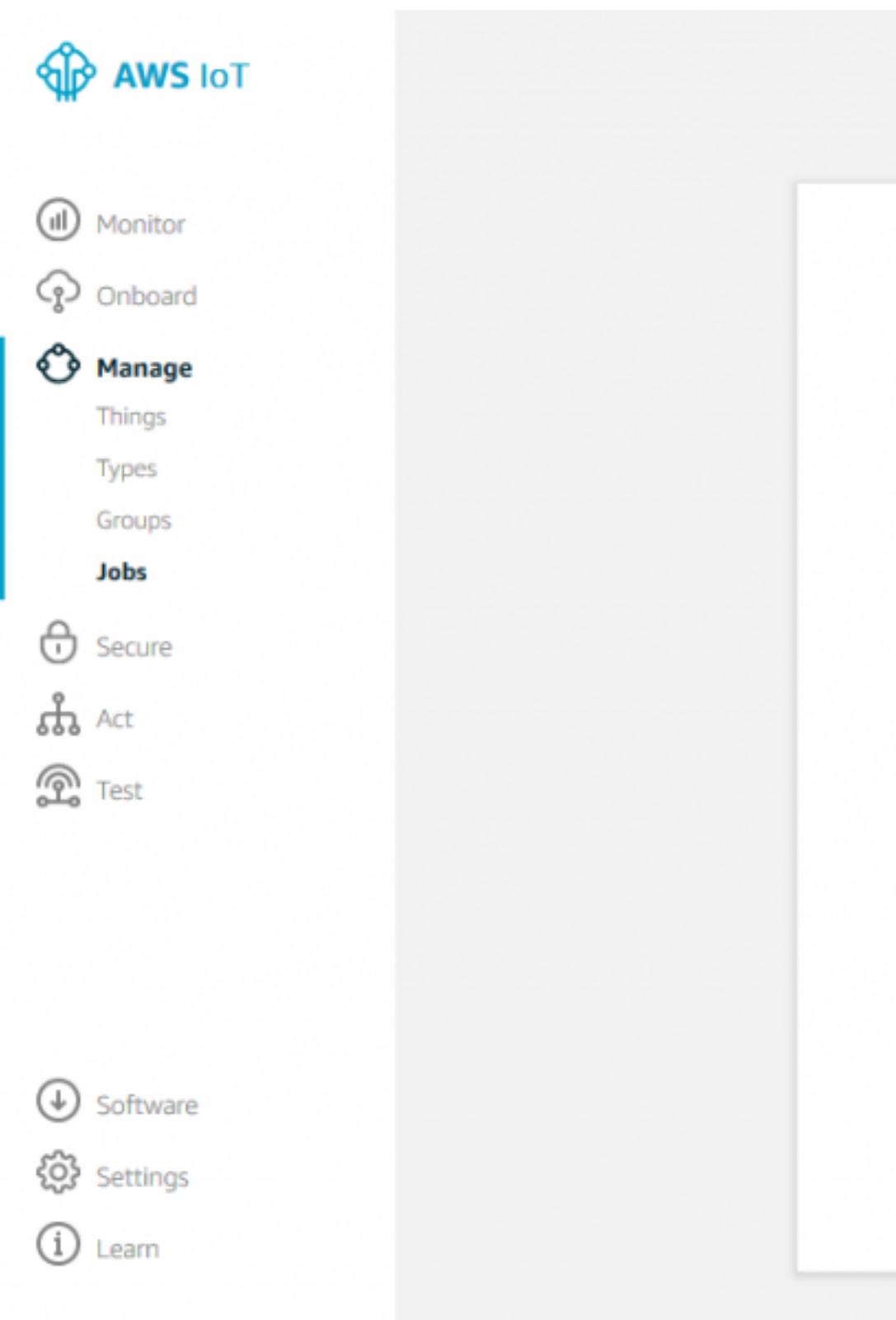
The following example shows how to use the [AWS IoT CLI](#) to create a job.

```
aws iot create-job \  
--job-id "example-job-01" \  
--targets "arn:aws:iot:::thing/MyRaspberryPi" \  
--document file:///example-job.json \  
--description "My First test job" \  
--target-selection SNAPSHOT
```

If you store your job document in an S3 bucket, use the `document-source` parameter instead of the `document` parameter to specify the Amazon S3 URL for the job document.

If you prefer to use the AWS IoT console, follow these steps to create a job.

1. Upload the job document to an S3 bucket. For information, see [How Do I Upload Files and Folders to an S3 Bucket?](#) in the *Amazon Simple Storage Service Console User Guide*.
2. In the AWS IoT console, choose **Manage**, and then choose **Jobs**.
3. Choose **Create a job**.



4. On the **Select a job** page, choose **Create custom job**.

The screenshot shows a modal window titled "CREATE JOB" with a large "X" button in the top-left corner. The main title is "Select a job". Below it is a detailed description of what jobs are: "AWS IoT Device Management job orchestration and notification set of remote operations called jobs that are sent to and executed on devices connected to AWS IoT." There are four listed options:

- Create a custom job**: "Send a request to acquire an executable job file from one of your devices connected to AWS IoT."
- Create an Amazon FreeRTOS Over-the-air (OTA) update job**: "This OTA update job will send your firmware image securely over-the-air to your FreeRTOS-based devices"
- Create a Greengrass Core update job**: "Create a snapshot job to update one or more Greengrass Core devices to a specific Greengrass Core or OTA agent version."

5. On the **Create a job** page, enter a unique job ID.

**Note**

We do not recommend using personally identifiable information in your job ID.

Under **Select devices to update**, select the device that you connected to AWS IoT.

The screenshot shows the 'Create a job' page in the AWS IoT Developer Guide. At the top, there is a back button and the title 'CREATE JOB'. Below the title, the heading 'Create a job' is displayed. The first section is labeled 'Job ID' with an input field containing 'example-job-01', which is highlighted with a red border. The next section is labeled 'Description' with an empty input field. Below these sections, the heading 'Select devices to update' is shown, followed by the instruction 'Browse and select the devices you want to include in this job.' A message indicates '1 thing(s) and 0 thing group(s) selected.' The 'Things' tab is active, showing a search bar and a list item 'MyRaspberryPi' with a checked checkbox, also highlighted with a red border.

CREATE JOB

## Create a job

Job ID

example-job-01

Description

Select devices to update

Browse and select the devices you want to include in this job.

1 thing(s) and 0 thing group(s) selected.

**Things** Thing Groups Summary

Search

MyRaspberryPi

6. Scroll down to **Add a job file** and choose the job document file that you uploaded to Amazon S3. Under **Job type**, select **Your job will complete after deploying to the selected devices/groups (snapshot)**. (The other option, **Your job will continue deploying to any devices added to the selected groups (continuous)**, is for deploying a job to groups of devices as devices are added to each group.) Leave the **Job executions rollout configuration** unchanged. Choose **Create**.

**Add a job file**

Upload a job file that defines what your job should do.

`example-job.json`

**Pre-sign resource URLs**

For an extra layer of security, you can pre-sign URLs that refer to resources in your job file.

Cannot find pre-sign url placeholder in the job file. Skip pre-sign configuration.

**Job type**

A job can run on the devices and/or groups selected, or remain open until completed.

Your job will complete after deploying to the selected devices

Your job will continue deploying to any devices added to the job

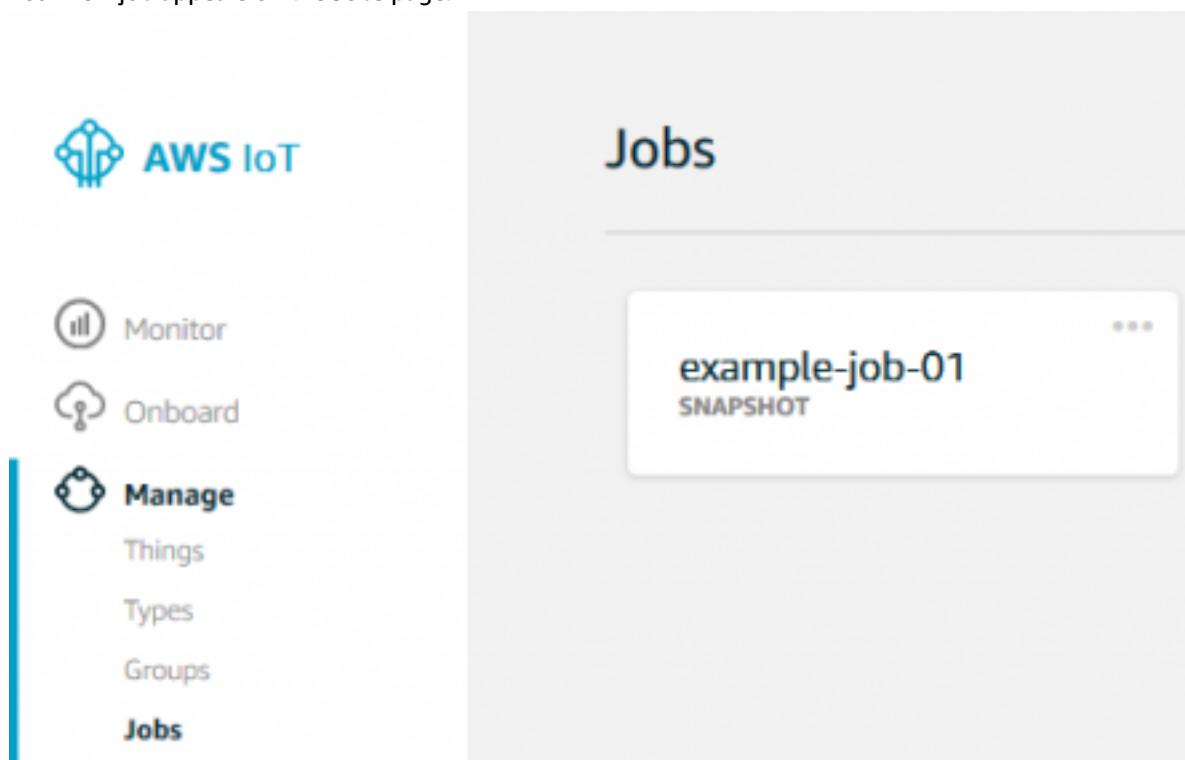
**Job executions rollout configuration**

Specify how quickly devices will be notified of a pending job execution.

**Maximum per minute (1-1000)**

1000

7. Your new job appears on the **Jobs** page.



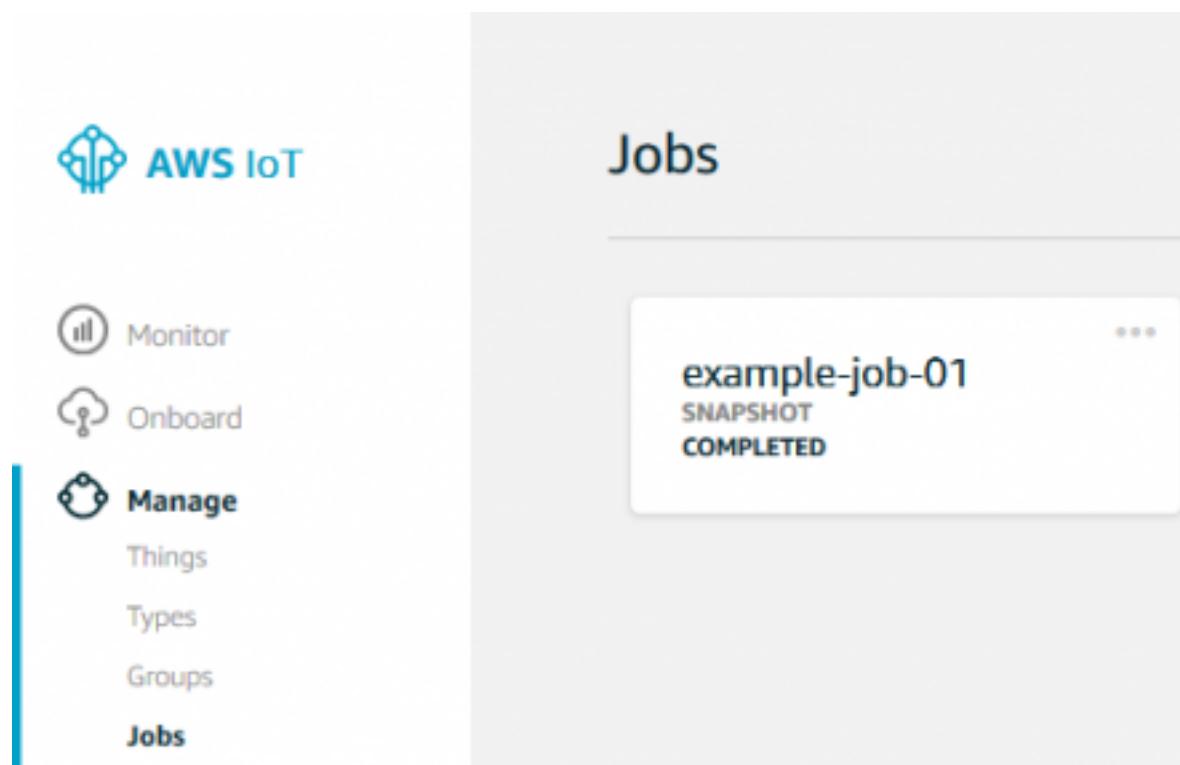
For more information about creating and deploying jobs, see [AWS IoT Core Jobs](#).

## Execute the Job on a Device

After your job is created, the Jobs service sends a notification of a pending job to your device. Your device gets the job details and job document through the [NextJobExecutionChanged](#) API. The `jobs-example.js` sample that you've already run executes the job on the device. When the job is complete, the sample publishes its completed status by using the [UpdateJobExecution](#) API. When you run the sample on your device, you see the following output.

```
node examples/jobs-example.js -f ./certs -F config.json
connect
startJobNotifications completed for thing: MyRaspberryPi
customJob operation handler invoked, jobId: example-job-01
```

When you refresh the **Jobs** page, you can see that your job was completed successfully.



## Tracking Job Progress with Job and Job Execution Events

You can use `Job` events and `JobExecution` events to track the progress of your job.

This is a helpful way to alert users, system administrators, and others that a job is complete or that a job execution has changed its status. For example, you can alert a user about a firmware update on a device or inform a system administrator about an issue in the device fleet that must be investigated and resolved.

Job events for the job in this example are published to the following topics when the job is completed or canceled.

```
$aws/events/job/example-job-01/completed  
$aws/events/job/example-job-01/canceled
```

Job execution events for the job in this example are sent to the following topics when the job execution reaches one of the possible final statuses.

```
$aws/events/jobExecution/example-job-01/succeeded  
$aws/events/jobExecution/example-job-01/failed  
$aws/events/jobExecution/example-job-01/rejected  
$aws/events/jobExecution/example-job-01/canceled  
$aws/events/jobExecution/example-job-01/removed
```

When the job execution on your device succeeds, AWS IoT Core publishes a [JobExecution succeeded](#) event. You can see this event in the console by navigating to the **Test** page and subscribing to the `$aws/events/jobExecution/example-job-01/succeeded` topic in the MQTT client.

The screenshot shows two side-by-side interfaces. On the left is the AWS IoT Test page, which lists several navigation options: Monitor, Onboard, Manage, Greengrass, Secure, Act, and Test. The 'Test' option is highlighted with a red box. On the right is an 'MQTT client' interface with a 'Subscriptions' section containing 'Subscribe to a topic' and 'Publish to a topic' buttons. A red box highlights the topic path '\$aws/e...' in the 'Publish to a topic' section, indicating where to subscribe to receive the success message.

The following message appears when the job execution for your device has completed successfully.

## Publish

Specify a topic and a message to publish with a QoS of 0.

```
$aws/events/jobExecution/example-job-01/succeeded
```

```
1  [
2    "message": "Hello from AWS IoT console"
3  ]
```

\$aws/events/jobExecution/example-job-01/...

Dec 19, 2017

```
{
  "eventType": "JOB_EXECUTION",
  "eventId": "af479061-a800-4d1f-a557-bbcd71243f7e",
  "timestamp": "1513709703",
  "operation": "succeeded",
  "jobId": "example-job-01",
  "thingArn": "arn:aws:iot:us-east-1:xxxxxxxxxxxx:thing",
  "status": "SUCCEEDED",
  "statusDetails": {
    "key": "value"
  }
}
```

AWS IoT Core also publishes a completed job event. You can see this event by subscribing to the \$aws/events/job/example-job-01/completed topic in the MQTT client.

## Publish

Specify a topic and a message to publish with a QoS of 0.

```
$aws/events/job/example-job-01/completed
```

```
1  {
2    "message": "Hello from AWS IoT console"
3 }
```

**\$aws/events/job/example-job-01/completed**

Dec 19, 2017

```
{
  "eventType": "JOB",
  "eventId": "a1817d74-0fld-42b3-b03f-acfc90af41e",
  "timestamp": "1513709645",
  "operation": "completed",
  "jobId": "example-job-01",
  "status": "COMPLETED",
  "targetSelection": "SNAPSHOT",
  "targets": [
    "arn:aws:iot:us-east-1:[REDACTED]:thing/MyRaspbe"
  ],
  "description": "Job example-job-01 for Thing MyRaspb",
  "completedAt": "1513709645105",
  "createdAt": "1513709615488",
  "lastUpdatedAt": "1513709645105",
  "jobProgressDetails": {
    "numberOfCanceledThings": 0,
    "numberOfRejectedThings": 0,
    "numberOfFailedThings": 0,
    "numberOfRemovedThings": 0,
    "numberOfSucceededThings": 1
  }
}
```

# AWS IoT Rules Tutorials

The following tutorials show you how to create and test AWS IoT rules. Before you begin, be sure to complete the [AWS IoT Getting Started Tutorial \(p. 5\)](#). It shows you how to create an AWS account and register a device in AWS IoT, which are prerequisites for these tutorials.

The scenario in this tutorial is a greenhouse with rows of plants. Each plant has a moisture sensor. At a predetermined interval, the moisture sensor sends its data to AWS IoT. The AWS IoT rules engine receives this data and writes it to a DynamoDB table. You create a rule to write data to DynamoDB and emulate the sensors using the AWS IoT MQTT client.

An AWS IoT rule consists of an 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 that matches the topic filter is received on a topic. 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 \(p. 267\)](#).

In these tutorials, we assume that you're using the AWS IoT MQTT client and that you are using my/greenhouse as the topic filter in the rules.

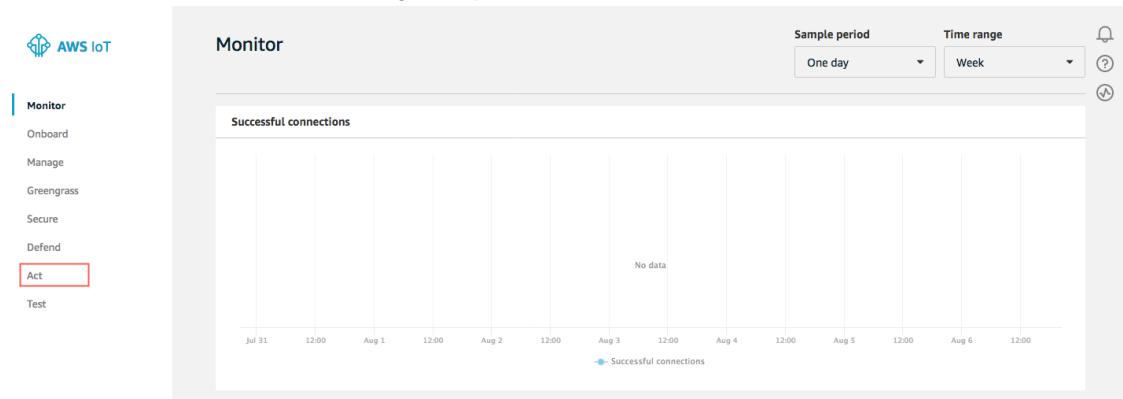
You can also use your own device, but you must know on which MQTT topic your device publishes so you can specify it as the topic filter in the rule. For more information, see [AWS IoT Rules \(p. 260\)](#).

## Creating a Rule with a DynamoDB Action

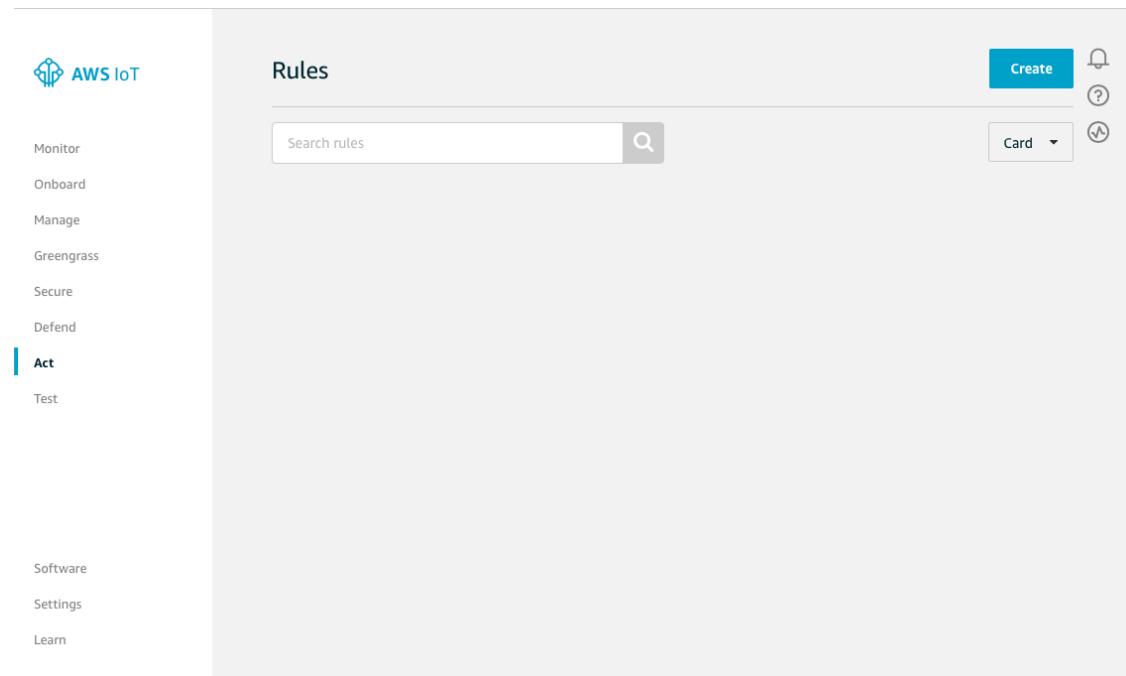
The DynamoDB action allows you to take information from an incoming MQTT message and write it to a DynamoDB table.

### To create a DynamoDB rule

1. In the [AWS IoT console](#), in the navigation pane, choose **Act**.



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



3. On the **Create a rule** page, enter a name and description for your rule.

**Note**

We do not recommend the use of personally identifiable information in rule names or descriptions.

### Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name  
GreenhouseRule

Description  
A DynamoDB rule for a greenhouse

4. Under **Rule query statement**, choose the latest version from the **Using SQL version** list. For **Rule query statement**, enter:

```
SELECT * FROM 'my/greenhouse'
```

("SELECT \*" specifies that you want to send the entire MQTT message that triggered the rule. "FROM 'my/greenhouse'" tells the rules engine to trigger this rule when an MQTT message whose topic matches this topic filter is received. Choose **Add action**.

**Rule query statement**

Indicate the source of the messages you want to process with this rule.

Using SQL version

2016-03-23 ▾

**Rule query statement**

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

1 SELECT \* FROM 'my/greenhouse'

**Set one or more actions**

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (\*.required)

Add action

5. On **Select an action**, choose **Insert a message into a DynamoDB table**, and then choose **Configure action**.

Select an action

Select an action.

-  Insert a message into a DynamoDB table  
DYNAMODB
-  Split message into multiple columns of a DynamoDB table (DynamoDBv2)  
DYNAMODBV2
-  Send a message to a Lambda function  
LAMBDA
-  Send a message as an SNS push notification  
SNS
-  Send a message to an SQS queue  
SQS
-  Send a message to an Amazon Kinesis Stream  
AMAZON KINESIS
-  Republish a message to an AWS IoT topic  
AWS IOT REPUBLISH
-  Store a message in an Amazon S3 bucket  
S3
-  Send a message to an Amazon Kinesis Firehose stream  
AMAZON KINESIS FIREHOSE
-  Send message data to CloudWatch  
CLOUDWATCH METRICS
-  Change the state of a CloudWatch alarm  
CLOUDWATCH ALARMS
-  Send a message to the Amazon Elasticsearch Service  
AMAZON ELASTICSEARCH
-  Send a message to a Salesforce IoT Input Stream  
SALESFORCE IOT
-  Send a message to an IoT Analytics Channel  
IOT ANALYTICS
-  Start a Step Functions state machine execution  
STEP FUNCTIONS

[Cancel](#) Configure action

6. On **Configure action**, choose **Create a new resource**.

Configure action

 Insert a message into a DynamoDB table  
DYNAMODB

The table must contain Partition and Sort keys.

\*Table name  
Choose a resource ▾  Create a new resource

\*Partition key  
Required field does not exist

\*Partition key type  
Required field does not exist

\*Partition key value

Sort key  
Optional field does not exist

Sort key type  
Optional field does not exist

Sort key value

Write message data to this column

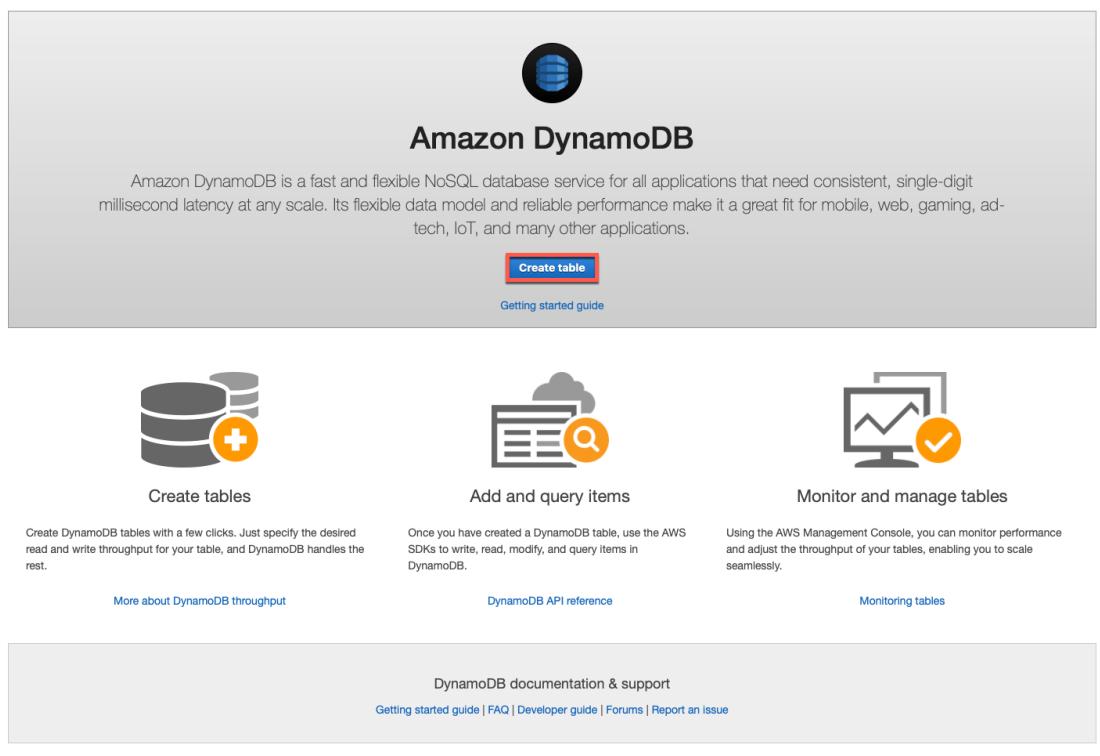
Operation 

Choose or create a role to grant AWS IoT access to perform this action.

No role selected 

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



8. On **Create DynamoDB table**, enter a name. In **Partition key**, enter **Row**. Select **Add sort key**, and then enter **PositionInRow** in the **Sort key** field. Row represents a row of plants in a greenhouse. PositionInRow represents the position of a plant in the row. Choose **String** for both the partition and sort keys, and then 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 don't close the tab, your DynamoDB table is not displayed in the **Table name** list on the **Configure action** page of the AWS IoT console.

**Create DynamoDB table**

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name*	GreenhouseTable	<small>i</small>
Primary key*	Partition key	
	Row	String <small>i</small>
	<input checked="" type="checkbox"/> Add sort key	
	PositionInRow	String <small>i</small>

**Table settings**

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".
- Encryption at Rest with DEFAULT encryption type NEW!

i You do not have the required role to enable Auto Scaling by default.  
Please refer to [documentation](#).

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

Cancel **Create**

9. On **Configure action**, choose your new table from the **Table name** list. In **Partition key value**, enter **#{row}**. This instructs the rule to take the value of the **row** attribute from the MQTT message

and write it into the **Row** column in the DynamoDB table. In **Sort key value**, enter `#{pos}`. This writes the value of the pos attribute into the **PositionInRow** column. In **Write message data to this column**, enter **Payload**. This inserts the message payload into the **Payload** column. Leave **Operation** blank. This field allows you to specify which operation (INSERT, UPDATE, or DELETE) you want to perform when the action is triggered. Choose **Create a new role**.

Configure action

Insert a message into a DynamoDB table  
DYNAMODB

The table must contain Partition and Sort keys.

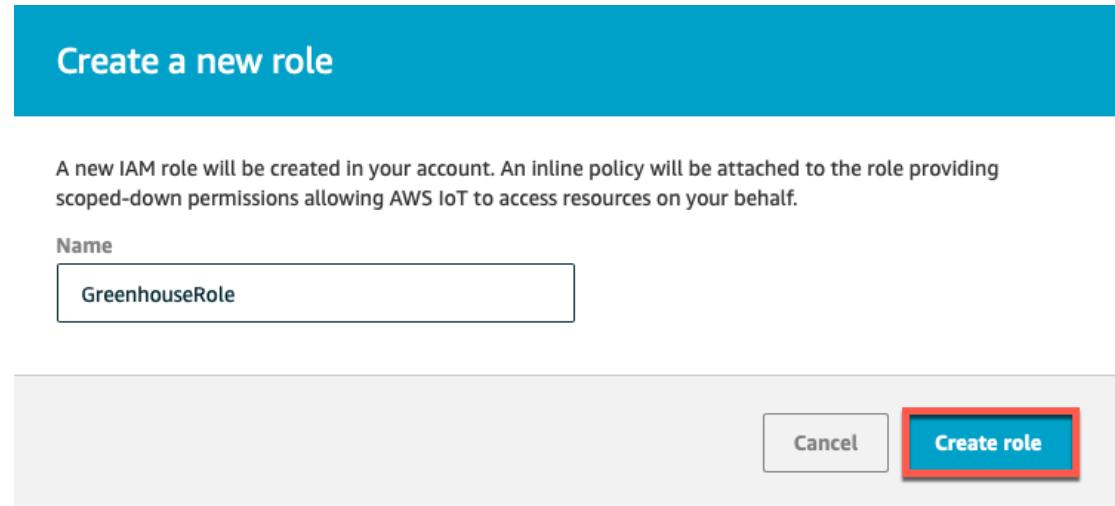
\*Table name  
GreenhouseTable

*Partition key Row	*Partition key type STRING	*Partition key value #{row}
Sort key PositionInRow	Sort key type STRING	Sort key value #{pos}
Write message data to this column Payload		
Operation ?		

Choose or create a role to grant AWS IoT access to perform this action.

No role selected

10. In **Create a new role**, enter a unique name, and then choose **Create role**.



11. Choose **Add action**.

### Configure action

 Insert a message into a DynamoDB table  
DYNAMODB

The table must contain Partition and Sort keys.

\*Table name  
 ⟳ Create a new resource

*Partition key <input type="text" value="Row"/>	*Partition key type <input type="text" value="STRING"/>	*Partition key value <input type="text" value="\${row}"/>
Sort key <input type="text" value="PositionInRow"/>	Sort key type <input type="text" value="STRING"/>	Sort key value <input type="text" value="\${pos}"/>
Write message data to this column <input type="text"/>		
Operation <span>?</span> <input type="text"/>		

Choose or create a role to grant AWS IoT access to perform this action.

Policy Attached ✓ Create Role Select

Cancel Add action

12. Choose **Create rule**.

## Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name

Description

---

**Rule query statement**  
Indicate the source of the messages you want to process with this rule.

Using SQL version

Rule query statement  
SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

```
1 SELECT * FROM 'my/greenhouse'
```

---

**Set one or more actions**  
Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (\*.required)

 **Insert a message into a DynamoDB table**  
GreenhouseTable Remove Edit ▾

**Add action**

---

**Error action**  
Optionally set an action that will be executed when something goes wrong with processing your rule.

**Add action**

---

**Tags**  
Apply tags to your resources to help organize and identify them. A tag consists of a case-sensitive key-value pair. [Learn more](#) about tagging your AWS resources.

Tag name  Value  **Clear**

**Add another**

---

47

**Cancel** **Create rule**

## Testing a Rule with a DynamoDB Action

1. To test the rule, open the AWS IoT console and from the navigation pane, choose **Test**.
2. Choose **Publish to a topic**. In the **Publish** section, enter **my/greenhouse**. In the message area, enter the following JSON:

```
{  
    "row" : "0",  
    "pos" : "0",  
    "moisture" : "75"  
}
```

The screenshot shows the AWS IoT MQTT client interface. At the top, it says "Connected as iotconsole-". The main area is titled "MQTT client" with a "Subscriptions" tab selected. Below it, there are two sections: "Subscribe to a topic" and "Publish to a topic". The "Publish to a topic" section has a "Subscription topic" input field containing "Specify a topic to subscribe to, e.g. myTopic/1" with a red box around it and the error message "This field is required.". It also has "Max message capture" set to 100 and "Quality of Service" options (0 or 1) with 0 selected. Under "MQTT payload display", the "Auto-format JSON payloads (improves readability)" option is selected. The "Publish" section below has a "Topic" input field containing "my/greenhouse" and a "Message" input field containing the JSON message from above. A red box highlights the "Publish to topic" button.

Return to the DynamoDB console and choose **Tables**.

The screenshot shows the AWS DynamoDB console. On the left, a sidebar has 'DynamoDB' selected, with 'Tables' highlighted by a red box. Other options include 'Dashboard', 'Backups', 'Reserved capacity', and 'Preferences'. Under 'DAX', there are 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main area is titled 'Create table' and contains a brief description of DynamoDB. A large blue 'Create table' button is at the top right. Below it is a section titled 'Recent alerts' with a note 'No CloudWatch alarms have been triggered.' and a link 'View all in CloudWatch'. A section titled 'Total capacity for US West (Oregon)' shows 'Provisioned read capacity' at 5 and 'Provisioned write capacity' at 5, with 'Reserved' columns at 0. Another section titled 'Service health' shows a single entry: 'Amazon DynamoDB (Oregon)' with a green checkmark, labeled 'Service is operating normally'. A link 'View complete service health details' is below.

Select the **GreenhouseTable**, and then choose **Items**. Your data is displayed on the **Items** tab.

The screenshot shows the 'Items' tab for the 'GreenhouseTable' in the AWS DynamoDB console. The left sidebar shows the 'Tables' section selected. The main area shows a table with one item. The item has a 'Name' field set to 'GreenhouseTable'. The 'payload' field contains the following JSON object:

```

{
  "moisture": {"S": "75"}, 
  "pos": {"S": "1"}, 
  "row": {"S": "0"}
}

```

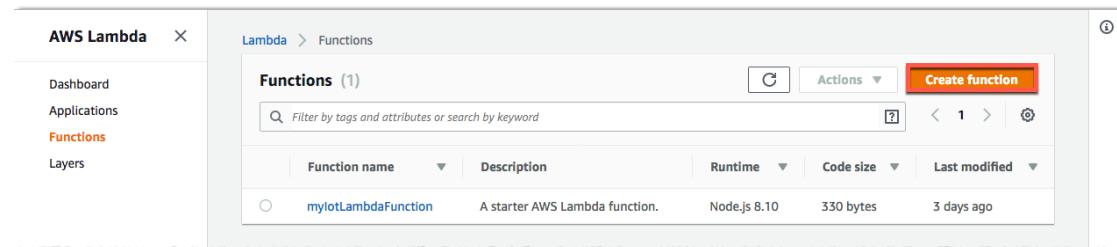
## Creating a Rule with a AWS Lambda Action

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 extract data from 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. Now you 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 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, you use the AWS IoT MQTT client to send a message that triggers the rule.

### Create a Lambda Function

1. In the [AWS Lambda console](#), choose **Create function**.



2. On the **Create function** page, choose **Use a blueprint**. In the **Blueprints** filter field, enter **hello-world**, and then press **Enter**. Choose the **hello-world-python** blueprint, and then choose **Configure**.

The screenshot shows the 'Create function' page under the 'Lambda > Functions' navigation. The title is 'Create function' with an 'Info' link. Below it, a sub-header says 'Choose one of the following options to create your function.' Three options are listed: 'Author from scratch' (radio button), 'Use a blueprint' (radio button, selected), and 'Browse serverless app repository' (radio button). The 'Use a blueprint' section includes a sub-section titled 'Blueprints' with an 'Info' link. A search bar at the top of this section contains the keyword 'hello-world'. Below the search bar, a table lists several blueprints, each with a preview icon, name, description, and runtime. The 'hello-world-python' blueprint is highlighted with a blue border. At the bottom right of the page are 'Cancel' and 'Configure' buttons, with 'Configure' being the one highlighted with a red box.

Blueprint	Description	Runtime
greengrass-hello-world	Deploy this lambda to a Greengrass core where it will send a hello world message to a topic	python · greengrass · iot · hello world
hello-world	A starter AWS Lambda function.	nodejs
<b>hello-world-python</b>	A starter AWS Lambda function.	python2.7
hello-world-python3	A starter AWS Lambda function.	python3.6
greengrass-hello-world-nodejs	Deploy this lambda to a Greengrass core where it will send a hello world message to a topic	nodejs6.10 · greengrass · iot · hello world

3. In **Basic information**, enter a name for your function.

**Note**

We do not recommend the use of personally identifiable information in rule names or descriptions.

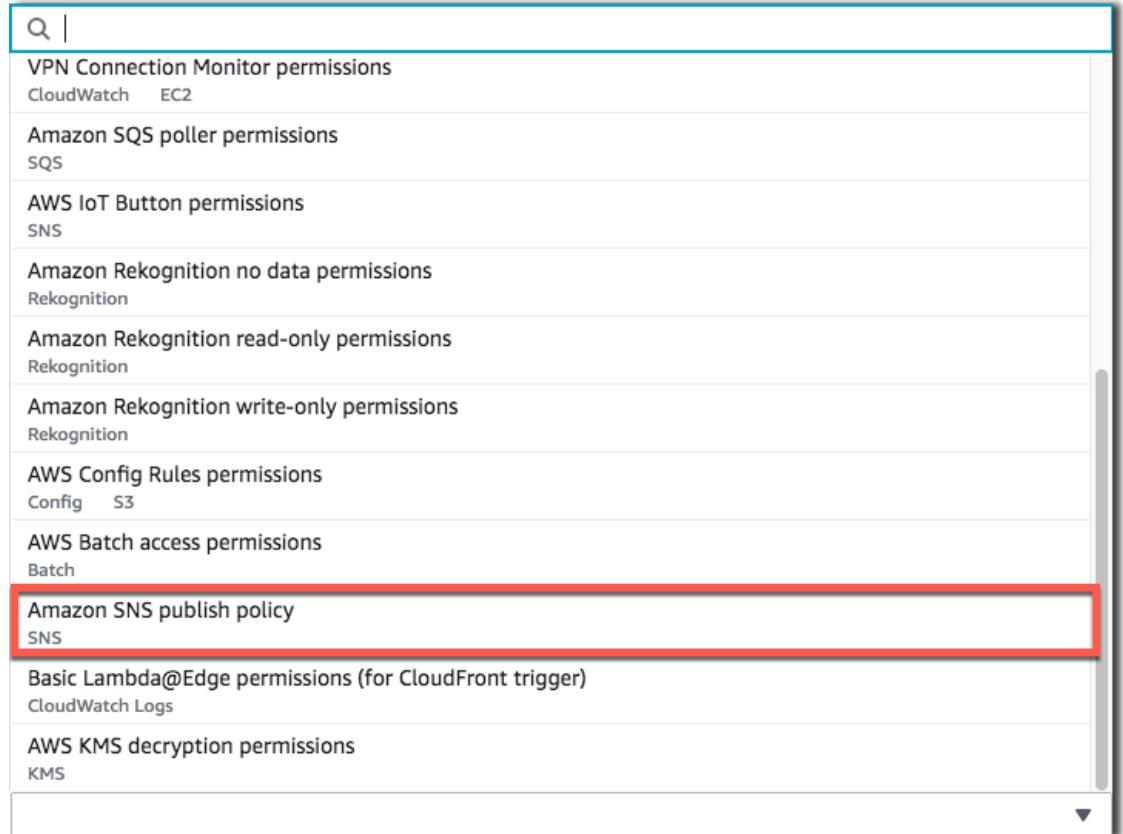
**Basic information** Info

Function name

Execution role  
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Existing role  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

- From **Execution Role**, choose **Create a new role from AWS policy templates**. Enter a name for the role. From **Policy templates**, choose **Amazon SNS publish policy**. Click outside of the drop-down menu to dismiss it.



VPN Connection Monitor permissions  
CloudWatch EC2

Amazon SQS poller permissions  
SQS

AWS IoT Button permissions  
SNS

Amazon Rekognition no data permissions  
Rekognition

Amazon Rekognition read-only permissions  
Rekognition

Amazon Rekognition write-only permissions  
Rekognition

AWS Config Rules permissions  
Config S3

AWS Batch access permissions  
Batch

**Amazon SNS publish policy**   
SNS

Basic Lambda@Edge permissions (for CloudFront trigger)  
CloudWatch Logs

AWS KMS decryption permissions  
KMS

- Choose **Create function**.

Basic information [Info](#)

Function name

myLambdaFunction

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Create a new role from AWS policy templates ▾

i Role creation might take a few minutes. The new role will be scoped to the current function. To use it with other functions, you can modify it in the IAM console.

Role name

Enter a name for your new role.

myiotLambdaFunction2-role

Use only letters, numbers, hyphens, or underscores with no spaces.

Policy templates [Info](#)

Choose one or more policy templates.

Amazon SNS publish policy X  
SNS

Lambda function code

Code is preconfigured by the chosen blueprint. You can configure it after you create the function.

Runtime

Python 2.7

```
1  from __future__ import print_function
2
3  import json
4
5  print('Loading function')
6
7
8  def lambda_handler(event, context):
9      #print("Received event: " + json.dumps(event, indent=2))
10     print("value1 = " + event['key1'])
11     print("value2 = " + event['key2'])
12     print("value3 = " + event['key3'])
13     return event['key1'] # Echo back the first key value
14     #raise Exception('Something went wrong')
15
```

\* These fields are required.

[Cancel](#)

[Previous](#)

[Create function](#)

- In the AWS Lambda console, choose the name of your Lambda function. Information about your Lambda function is displayed. Scroll down to the **Function code** section and replace the existing code with the following example.

```
from __future__ import print_function

import json
import boto3

print('Loading function')

def lambda_handler(event, context):

    # Parse the JSON message
    eventText = json.dumps(event)

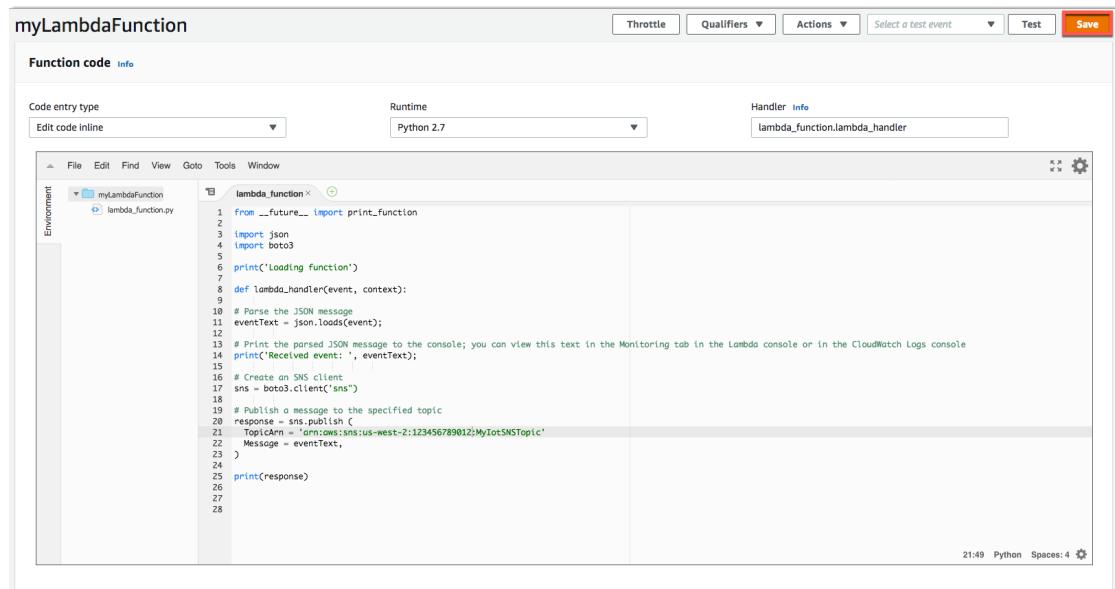
    # Print the parsed JSON message to the console. You can view this text in the Monitoring tab in the AWS Lambda console or in the Amazon CloudWatch Logs console.
    print('Received event: ', eventText)

    # Create an SNS client
    sns = boto3.client('sns')

    # Publish a message to the specified topic
    response = sns.publish (
        TopicArn = 'arn:aws:iam::123456789012:My_IoT_SNS_Topic',
        Message = eventText
    )

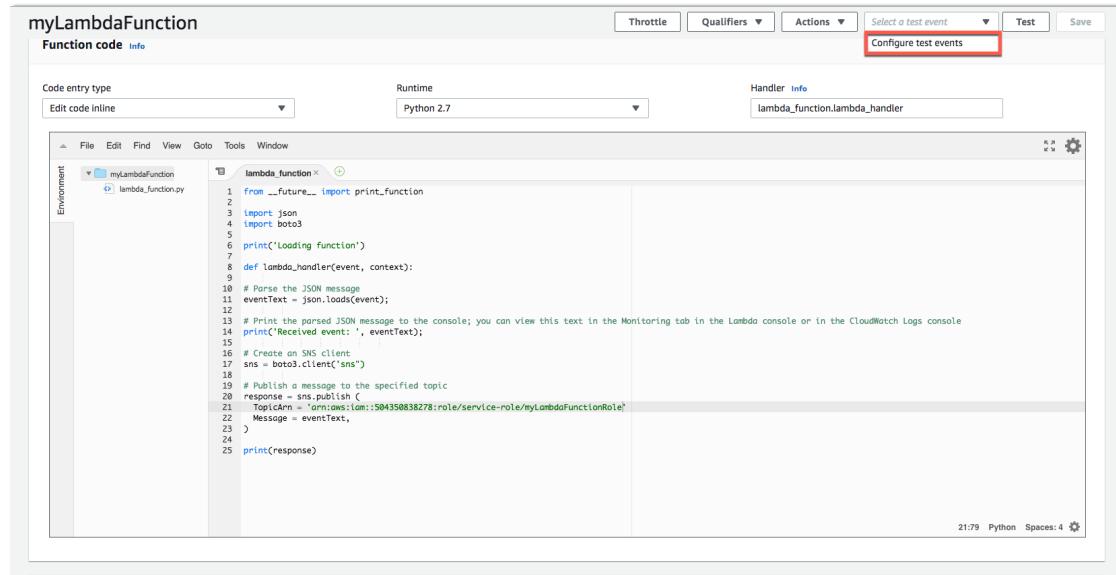
    print(response)
```

- Replace the value of `TopicArn` with the ARN of the Amazon SNS topic that you created in [Configure and Test Rules](#).
- Choose **Save**.



## Test Your Lambda Function

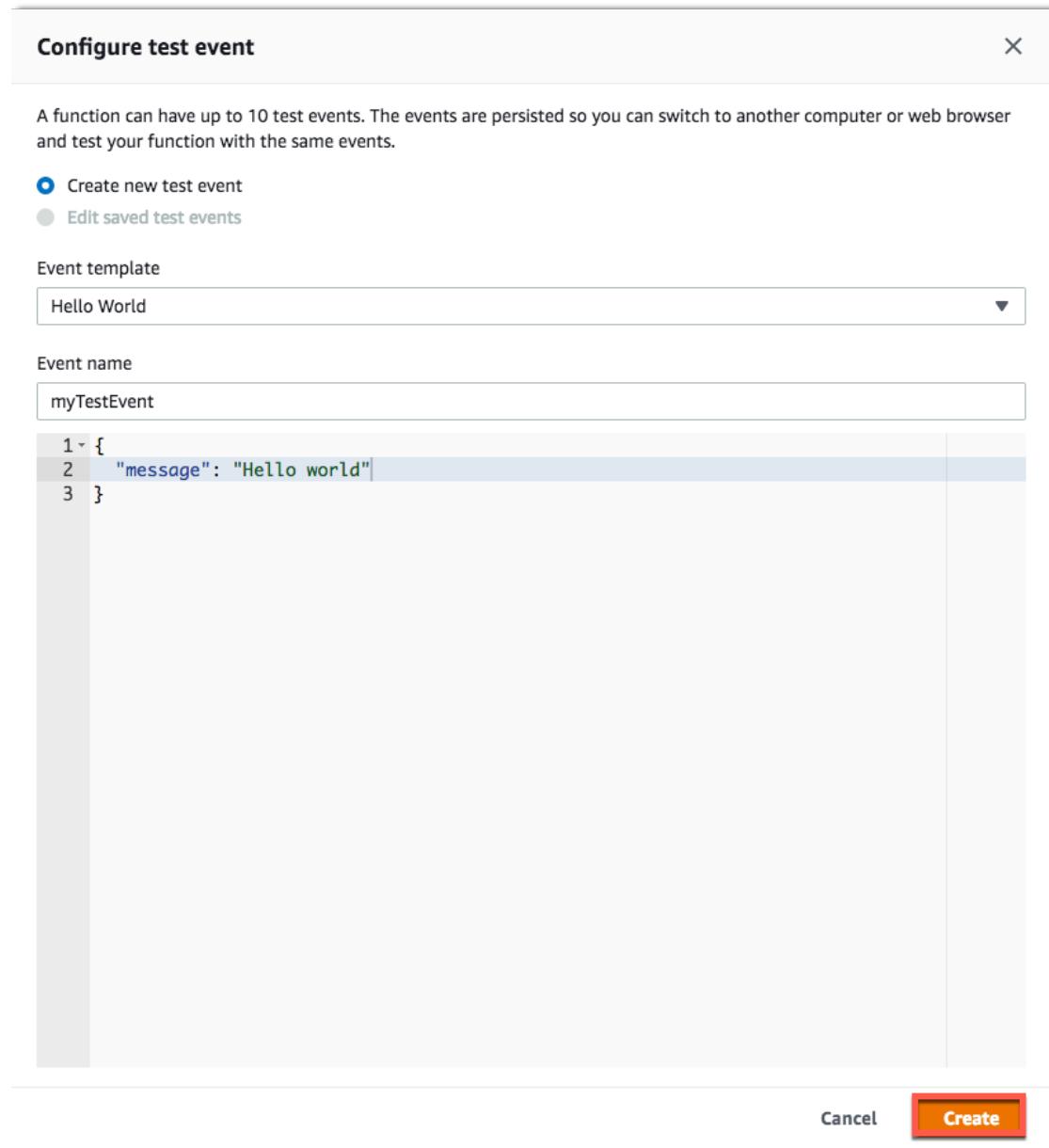
1. In the upper right of the Lambda function details page, from **Select a test event**, choose **Configure test events**.



2. On **Configure test event**, enter a name for your test event and replace the message JSON with the following:

```
{  
    "message" : "Hello, world"  
}
```

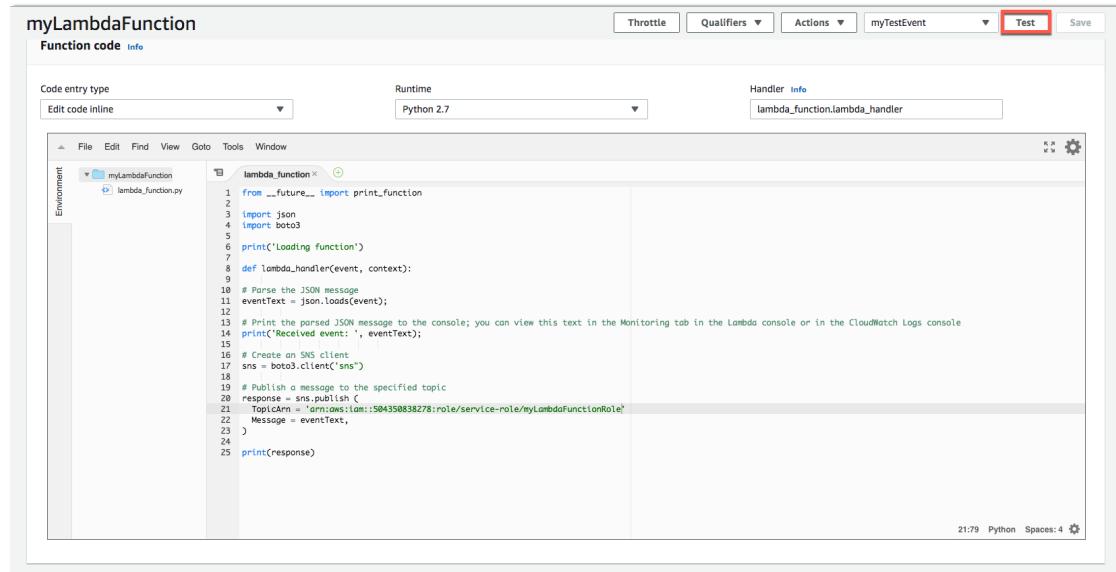
Choose **Create**.



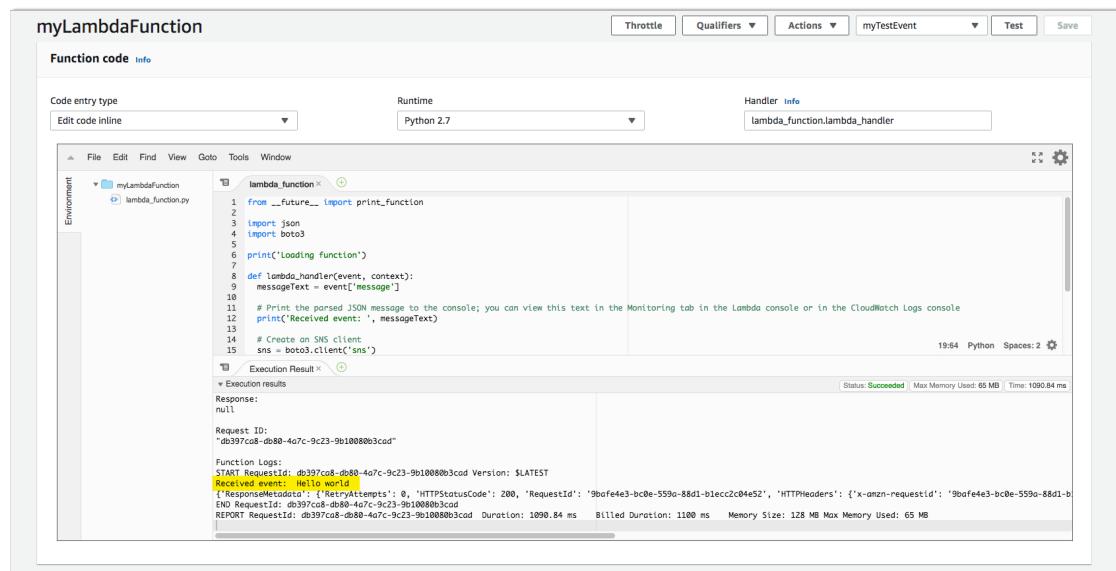
3. In the upper right of the Lambda function details page, choose **Test** to test your Lambda function with the message you specified in the test event.

## AWS IoT Developer Guide

### Create a Rule with a Lambda Action



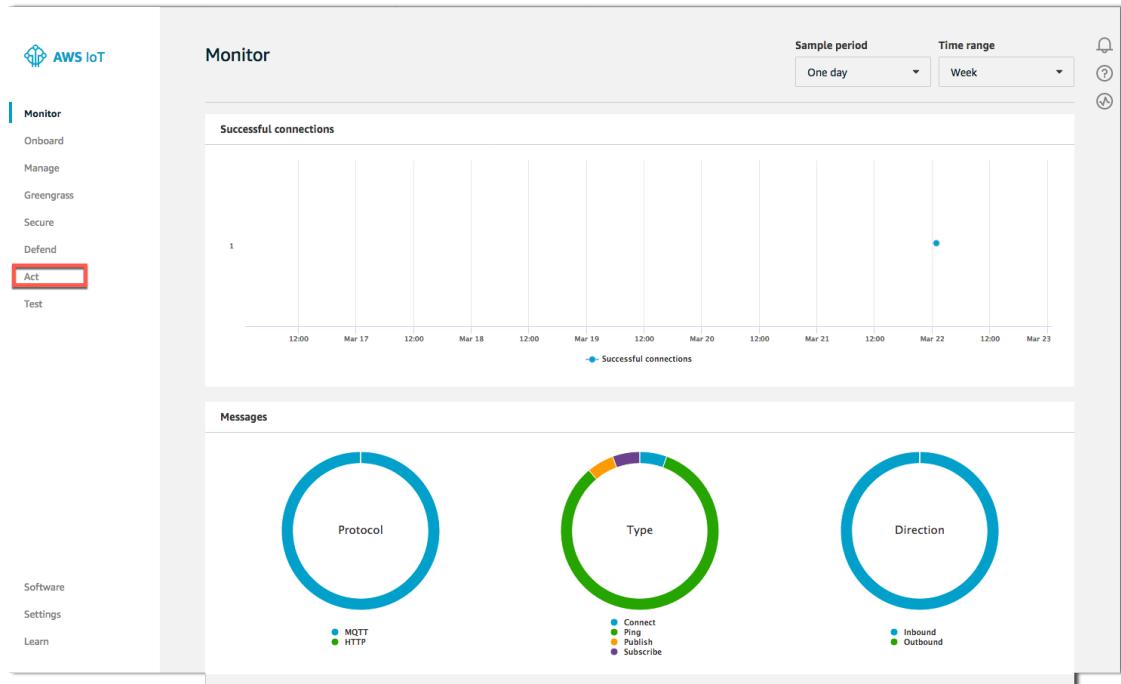
- Under your Lambda function code, on the **Execution result** tab, you see the output from the Lambda function.



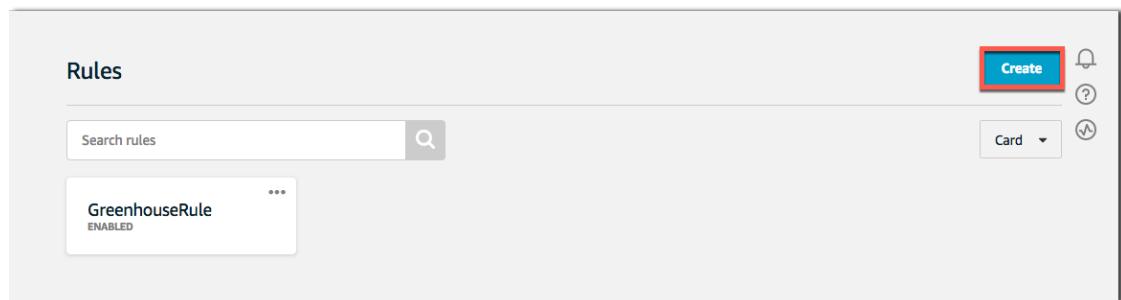
## Create a Rule with a Lambda Action

This section provides steps for creating a rule with a Lambda action and an error action. The Lambda action calls your Lambda function. If an error occurs when calling the Lambda function, the error action publishes a message to the `lambda/error` MQTT topic. This is useful when you are testing the rule.

- Browse to the AWS IoT console, and from the navigation pane, choose **Act**.



2. Choose **Create** to create an AWS IoT rule.



3. On the **Create a rule** page, enter a name for your rule.

The screenshot shows the 'Create a rule' page. The title 'Create a rule' is at the top in a blue header. Below it, a descriptive text states: 'Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).'. There are two input fields: 'Name' containing 'myLambdaRule' and 'Description' which is empty.

4. In **Rule query statement**, enter the following query:

```
SELECT * FROM "my/lambda/topic"
```

**Rule query statement**

Indicate the source of the messages you want to process with this rule.

Using SQL version

2016-03-23 ▾

**Rule query statement**

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

```
1 SELECT * FROM "my/lambda/topic"
```

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

**Set one or more actions**

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (\*.required)

**Add action**

6. Under **Select an action**, choose **Send a message to a Lambda function**, and then choose **Configure action**.

Select an action

Select an action.

<input type="radio"/>	 Insert a message into a DynamoDB table DYNAMODB
<input type="radio"/>	 Split message into multiple columns of a DynamoDB table (DynamoDBv2) DYNAMODBV2
<input checked="" type="radio"/>	 Send a message to a Lambda function LAMBDA
<input type="radio"/>	 Send a message as an SNS push notification SNS
<input type="radio"/>	 Send a message to an SQS queue SQS
<input type="radio"/>	 Send a message to an Amazon Kinesis Stream AMAZON KINESIS
<input type="radio"/>	 Republish a message to an AWS IoT topic AWS IOT REPUBLISH
<input type="radio"/>	 Store a message in an Amazon S3 bucket S3
<input type="radio"/>	 Send a message to an Amazon Kinesis Firehose stream AMAZON KINESIS FIREHOSE
<input type="radio"/>	 Send message data to CloudWatch CLOUDWATCH METRICS
<input type="radio"/>	 Change the state of a CloudWatch alarm CLOUDWATCH ALARMS
<input type="radio"/>	 Send a message to the Amazon Elasticsearch Service AMAZON ELASTICSEARCH
<input type="radio"/>	 Send a message to a Salesforce IoT Input Stream SALESFORCE IOT
<input type="radio"/>	 Send a message to an IoT Analytics Channel IOT ANALYTICS
<input type="radio"/>	 Start a Step Functions state machine execution STEP FUNCTIONS

[Cancel](#) Configure action

7. On **Configure action**, choose **Select**.

### Configure action

 Send a message to a Lambda function  
LAMBDA

We'll set [the permissions](#) on the Lambda function for you. [Create a new Lambda function](#)

\*Function name

No lambda function selected Select

[Cancel](#) Add action

8. Choose your Lambda function.

### Configure action

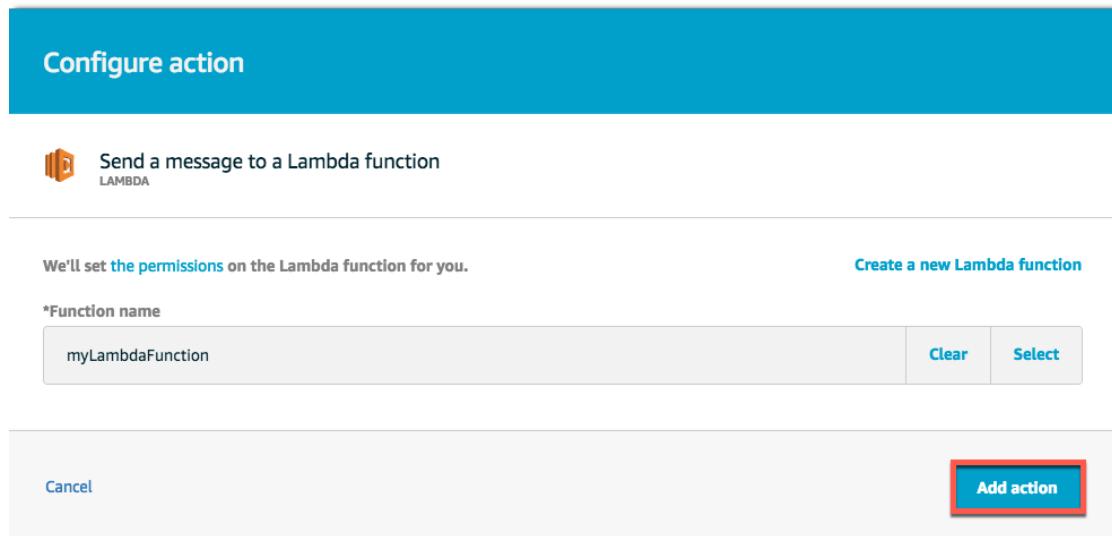
 Send a message to a Lambda function  
LAMBDA

We'll set [the permissions](#) on the Lambda function for you. [Create a new Lambda function](#)

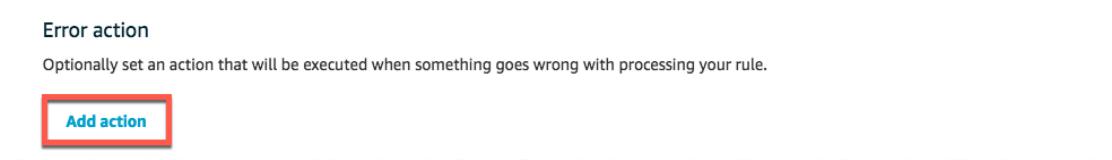
\*Function name

No lambda function selected	<a href="#">Refresh</a>	<a href="#">Close</a>
<input type="text" value="Search for lambda functions"/>		
myLambdaFunction	<span style="border: 2px solid red; padding: 2px;">Select</span>	
<a href="#">Cancel</a> <span style="background-color: #0070C0; color: white; padding: 5px 10px;">Add action</span>		

9. Choose **Add action**.



10. On the **Create a rule** page, in **Error action**, choose **Add action**.



11. On **Set an action as error action**, choose **Republish a message to an AWS IoT topic**, and then choose **Configure action**.

### Set an action as error action

Set an action as error action.

<input type="radio"/>	 Insert a message into a DynamoDB table DYNAMODB
<input type="radio"/>	 Split message into multiple columns of a DynamoDB table (DynamoDBv2) DYNAMODBV2
<input type="radio"/>	 Send a message to a Lambda function LAMBDA
<input type="radio"/>	 Send a message as an SNS push notification SNS
<input type="radio"/>	 Send a message to an SQS queue SQS
<input type="radio"/>	 Send a message to an Amazon Kinesis Stream AMAZON KINESIS
<input checked="" type="radio"/>	 Republish a message to an AWS IoT topic AWS IOT REPUBLISH
<input type="radio"/>	 Store a message in an Amazon S3 bucket S3
<input type="radio"/>	 Send a message to an Amazon Kinesis Firehose stream AMAZON KINESIS FIREHOSE
<input type="radio"/>	 Send message data to CloudWatch CLOUDWATCH METRICS
<input type="radio"/>	 Change the state of a CloudWatch alarm CLOUDWATCH ALARMS
<input type="radio"/>	 Send a message to the Amazon Elasticsearch Service AMAZON ELASTICSEARCH
<input type="radio"/>	 Send a message to a Salesforce IoT Input Stream SALESFORCE IOT
<input type="radio"/>	 Send a message to an IoT Analytics Channel IOT ANALYTICS
<input type="radio"/>	 Start a Step Functions state machine execution STEP FUNCTIONS

[Cancel](#) Configure action

12. On **Configure action**, under **Topic**, enter `lambda/error`.

### Configure action

 Republish a message to an AWS IoT topic  
AWS IOT REPUBLISH

This action will republish the message to another AWS IoT topic.

\*Topic [?](#)

Choose or create a role to grant AWS IoT access to perform this action.

No role selected [Create Role](#) [Select](#)

[Cancel](#) [Add action](#)

13. Under **Choose or create a role to grant AWS IoT access to perform this action**, choose **Create Role**.

### Configure action

 Republish a message to an AWS IoT topic  
AWS IOT REPUBLISH

This action will republish the message to another AWS IoT topic.

\*Topic [?](#)

Choose or create a role to grant AWS IoT access to perform this action.

No role selected [Create Role](#) [Select](#)

[Cancel](#) [Add action](#)

14. In **Create a new role**, enter a name for the role, and then choose **Create role**.

## Create a new role

A new IAM role will be created in your account. An inline policy will be attached to the role providing scoped-down permissions allowing AWS IoT to access resources on your behalf.

Name

15. In **Configure action**, choose **Add action**.

## Configure action

 Republish a message to an AWS IoT topic  
AWS IOT REPUBLISH

This action will republish the message to another AWS IoT topic.

\*Topic [?](#)

Choose or create a role to grant AWS IoT access to perform this action.

myLambdaErrorActionRole	Policy Attached ✓	<input type="button" value="Create Role"/>	<input type="button" value="Select"/>
-------------------------	-------------------	--	---------------------------------------

16. In **Create a rule**, choose **Create rule**.

## Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name

Description

### Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version

#### Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

```
1 SELECT * FROM "my/lambda/topic"
```

### Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (\*.required)



Send a message to a Lambda function  
myLambdaFunction

[Remove](#) [Edit](#)

[Add action](#)

### Error action

Optionally set an action that will be executed when something goes wrong with processing your rule.



Republish a message to an AWS IoT topic  
lambda/error

[Remove](#) [Edit](#)

### Tags

Apply tags to your resources to help organize and identify them. A tag consists of a case-sensitive key-value pair. [Learn more](#) about tagging your AWS resources.

Tag name

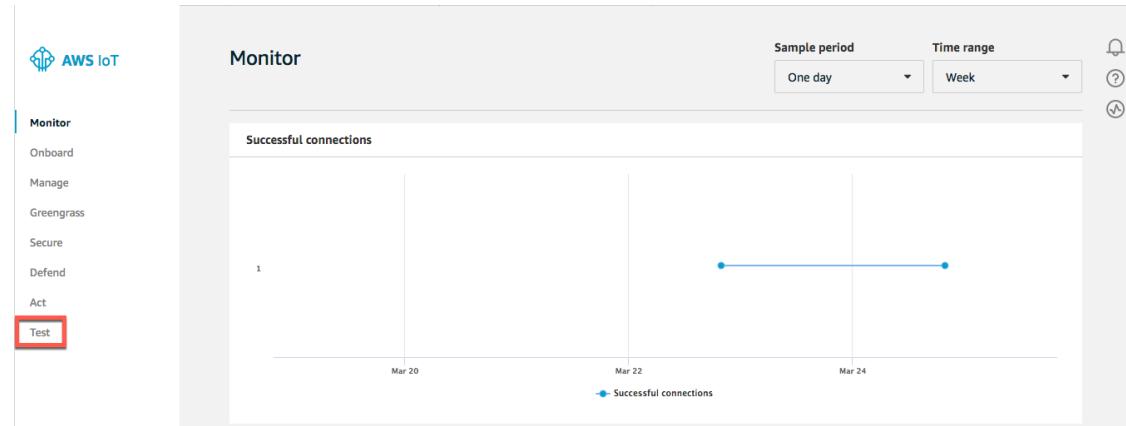
Value

[Clear](#)

[Add another](#)

## Test Your Rule with a Lambda Action

- To test your Lambda action, open the AWS IoT console, and from the navigation pane, choose **Test**.

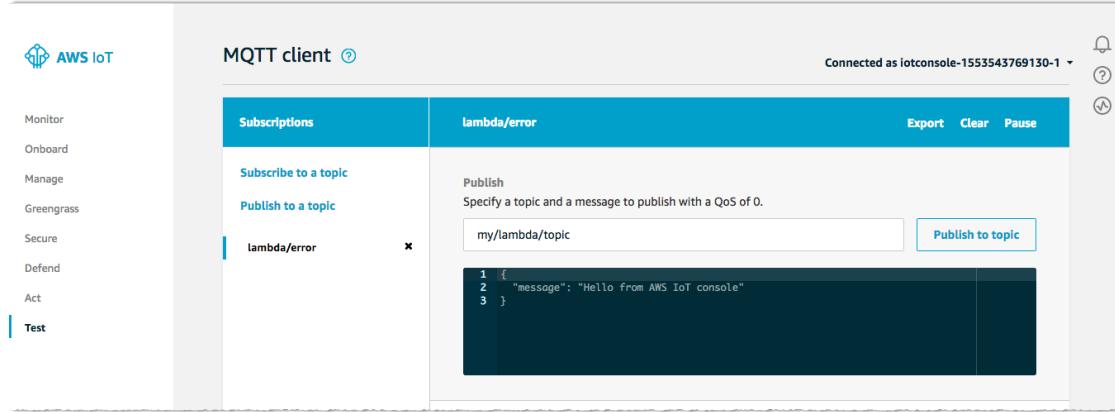


- In the MQTT client, under **Subscription topic**, enter `lambda/error`, and then choose **Subscribe to topic**.

The screenshot shows the AWS IoT Subscriptions page. On the left, a sidebar has 'Subscriptions' selected. The main area is titled 'Subscriptions' and contains two sections: 'Subscribe to a topic' and 'Publish to a topic'. Under 'Subscribe to a topic', there is a 'Subscription topic' input field containing `lambda/error`, and a 'Subscribe' button which is highlighted with a red box. Below this, there are settings for 'Max message capture' (set to 100) and 'Quality of Service' (radio button 0 is selected). Under 'MQTT payload display', radio button 0 ('Auto-format JSON payloads (improves readability)') is selected. The 'Publish' section below has a 'Specify a topic and a message to publish with a QoS of 0.' input field and a 'Publish to topic' button. A dark panel at the bottom displays a JSON message: 

```
1  {
2    "message": "Hello from AWS IoT console"
3  }
```

- Under **Publish**, enter `my/lambda/topic`, and then choose **Publish to topic** to publish the default JSON message.



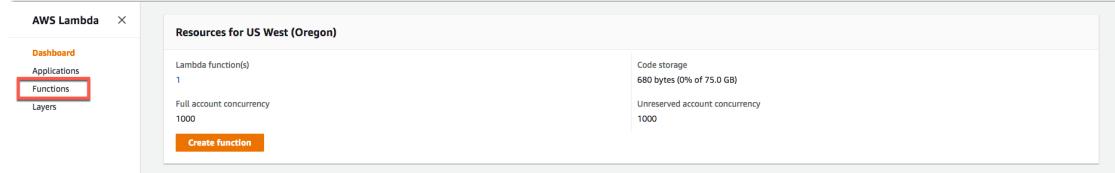
Publishing this message should trigger the rule and call your Lambda function. Your Lambda function pushes an Amazon SNS message to a phone number subscribed to your Amazon SNS topic. If you do not get a text message, in the MQTT client, check to see if any messages were published to `lambda/error`.

## Troubleshooting a Rule with a Lambda Action

If your Lambda function is called, but you do not receive a text message, make sure your phone number is subscribed to your Amazon SNS topic. If your phone number is subscribed, check the CloudWatch logs for your Lambda function. AWS Lambda writes logs to CloudWatch, which makes it possible for you to see output from your Lambda function.

### To view CloudWatch logs

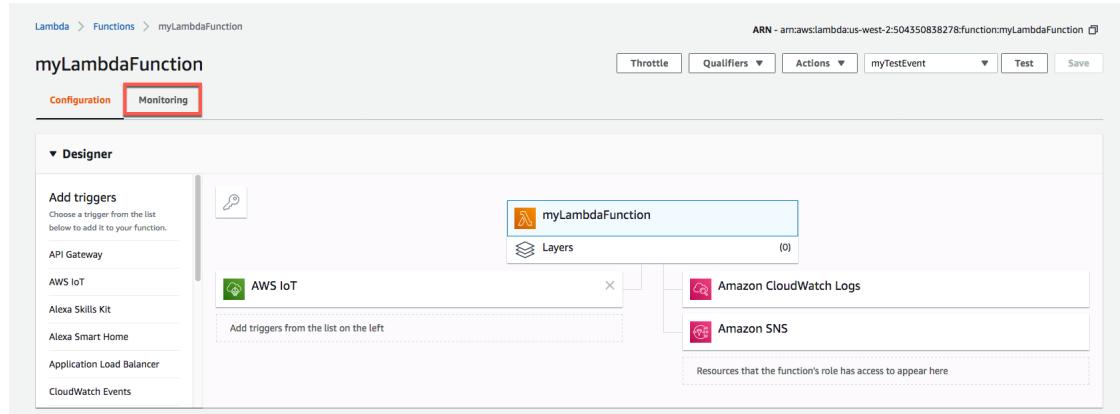
1. In the Lambda console, from the navigation pane, choose **Functions**.



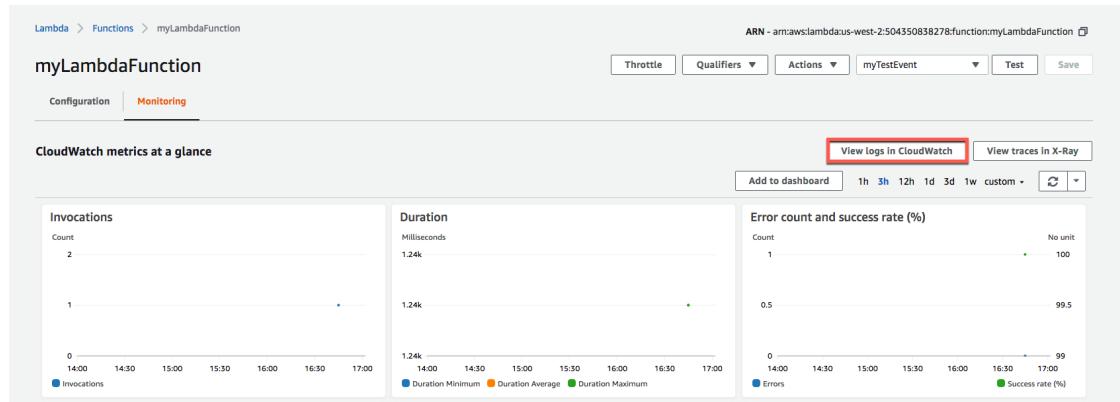
2. Choose your Lambda function.



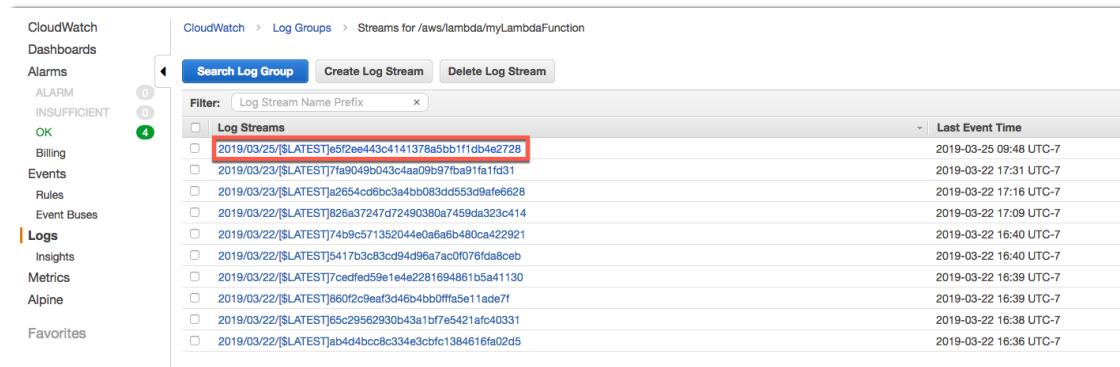
3. On the Lambda function details page, choose the **Monitoring** tab.



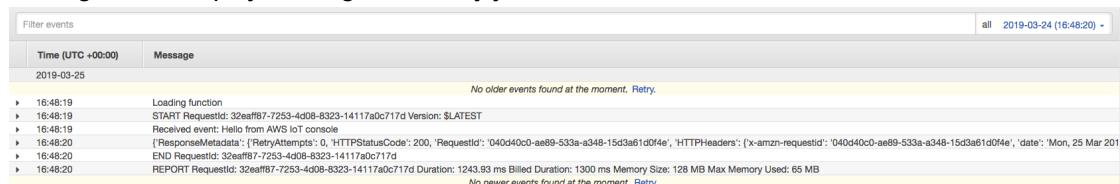
#### 4. Choose View logs in CloudWatch.



#### 5. Choose the latest log stream.



#### 6. The log stream displays the logs written by your Lambda function.



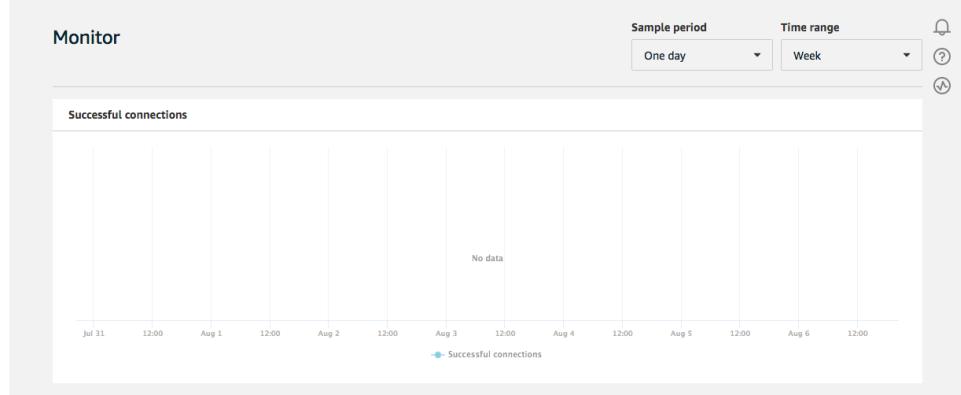
## Creating an Amazon SNS Rule

You can define a rule that sends message data to an Amazon SNS topic.

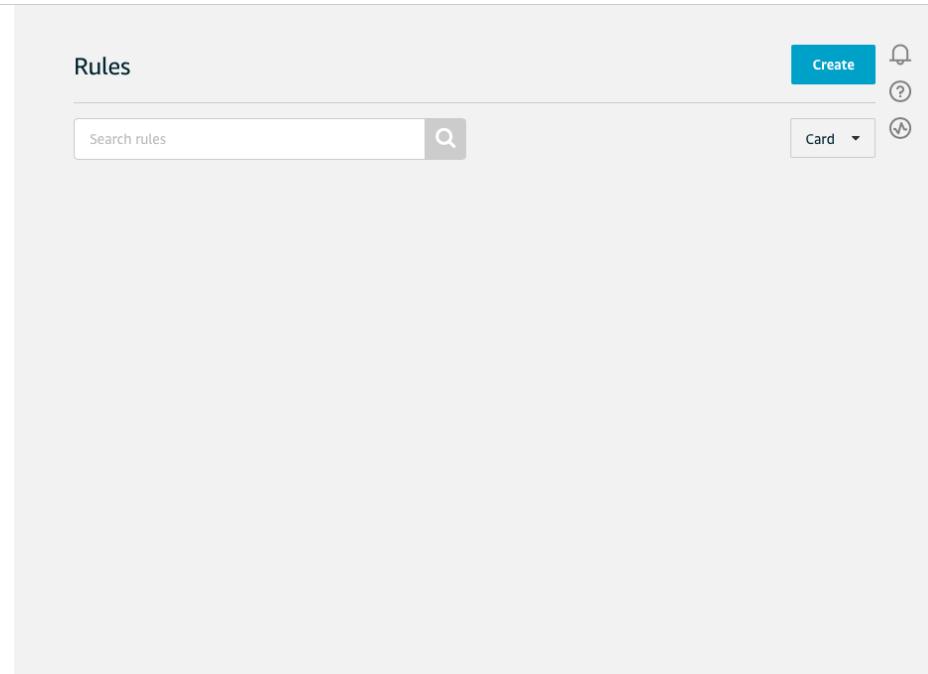
In this tutorial, you create a rule that sends the name of the AWS IoT thing that triggered the rule to all subscribers of an Amazon SNS topic.

### To create a rule with an SNS action

1. In the [AWS IoT console](#), in the navigation pane, choose **Act**.



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



3. Enter a name and brief description for your rule.

**Note**

We do not recommend the use of personally identifiable information in rule names or descriptions.

## Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name  
MySNSRule

Description  
A more complex SNS rule.

4. In the **Rule query statement** editor, enter the following:

```
SELECT *, topic(3) as thing FROM '$aws/things/+/shadow/update/accepted'
```

(The topic filter following the "FROM" specifies the topics that trigger the rule's action when a message is published to them. The plus sign (+) used in the topic filter is a wildcard character that matches any thing name. The "topic(3)" attribute following "SELECT" appends the thing name, which is the third topic field, onto the message contents.)

### Rule query statement

Indicate the source of the messages you want to process with this rule.

#### Using SQL version

2016-03-23 ▾

### Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

```
1 SELECT *, topic(3) as thing FROM '$aws/things/+/shadow/update/accepted'
```

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

### Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (\*.required)

Add action

6. Under **Select an action**, choose **Send a message as an SNS push notification**, and then choose **Configure action**.

### Select an action

Select an action.

<input type="radio"/>	 Insert a message into a DynamoDB table DYNAMODB
<input type="radio"/>	 Split message into multiple columns of a DynamoDB table (DynamoDBv2) DYNAMODBV2
<input type="radio"/>	 Send a message to a Lambda function LAMBDA
<input checked="" type="radio"/>	 Send a message as an SNS push notification SNS
<input type="radio"/>	 Send a message to an SQS queue SQS
<input type="radio"/>	 Send a message to an Amazon Kinesis Stream AMAZON KINESIS
<input type="radio"/>	 Republish a message to an AWS IoT topic AWS IOT REPUBLISH
<input type="radio"/>	 Store a message in an Amazon S3 bucket S3
<input type="radio"/>	 Send a message to an Amazon Kinesis Firehose stream AMAZON KINESIS FIREHOSE
<input type="radio"/>	 Send message data to CloudWatch CLOUDWATCH METRICS
<input type="radio"/>	 Change the state of a CloudWatch alarm CLOUDWATCH ALARMS
<input type="radio"/>	 Send a message to the Amazon Elasticsearch Service AMAZON ELASTICSEARCH
<input type="radio"/>	 Send a message to a Salesforce IoT Input Stream SALESFORCE IOT
<input type="radio"/>	 Send a message to IoT Analytics IOT ANALYTICS
<input type="radio"/>	 Send a message to an IoT Events Input IOT EVENTS
<input type="radio"/>	 Start a Step Functions state machine execution STEP FUNCTIONS

[Cancel](#) [Configure action](#)

7. On the **Configure action** page, for **SNS target**, choose **Create**.

The screenshot shows the 'Configure action' page for creating an Amazon SNS rule. At the top, there's a title bar with the text 'Configure action'. Below it, a section titled 'Send a message as an SNS push notification' includes an 'SNS' icon. Underneath, a field labeled 'SNS target' shows 'No topic selected' and has two buttons: 'Create' (which is highlighted with a red box) and 'Select'. A dropdown menu for 'Message format' is set to 'Select'. A note below says 'Choose or create a role to grant AWS IoT access to perform this action.' with 'No role selected' and 'Select' buttons. At the bottom, there's a 'Create' button, a 'Name' field containing 'MySNSTopic', and a 'Cancel' button next to a 'Create' button.

Configure action

Send a message as an SNS push notification  
SNS

\*SNS target

No topic selected Create Select

Message format

Select ▾

Choose or create a role to grant AWS IoT access to perform this action.

No role selected Select

Cancel Add action

**Create**

Name

MySNSTopic

Cancel Create

8. Enter a topic name in the dialog box, and then choose **Create**.
9. On the **Configure action** page, for **SNS target**, choose the SNS topic you just created. For **Message format**, choose **RAW**. Under **Choose or create a role to grant AWS IoT access to perform this action**, choose **Create Role**.

### Configure action

 Send a message as an SNS push notification  
SNS

\*SNS target

MySNSTopic	<a href="#">Create</a>	<a href="#">Clear</a>	<a href="#">Select</a>
------------	------------------------	-----------------------	------------------------

Message format

RAW
-----

Choose or create a role to grant AWS IoT access to perform this action.

No role selected	<a href="#">Create Role</a>	<a href="#">Select</a>
------------------	-----------------------------	------------------------

[Cancel](#) [Add action](#)

10. Enter a name for the role, and then choose **Create role**.

### Create a new role

A new IAM role will be created in your account. An inline policy will be attached to the role providing scoped-down permissions allowing AWS IoT to access resources on your behalf.

Name

MySNSRole
-----------

[Cancel](#) [Create role](#)

11. In **Configure action**, choose **Add action**.

### Configure action

 Send a message as an SNS push notification  
SNS

\*SNS target

MySNSTopic	<a href="#">Create</a>	<a href="#">Clear</a>	<a href="#">Select</a>
------------	------------------------	-----------------------	------------------------

Message format

RAW	▼
-----	---

Choose or create a role to grant AWS IoT access to perform this action.

MySnsRole	Policy Attached ✓	<a href="#">Create Role</a>	<a href="#">Select</a>
-----------	-------------------	-----------------------------	------------------------

[Cancel](#) Add action

12. Choose **Create rule**.

## Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

**Name**  
MySNSRule

**Description**  
A more complex SNS rule.

---

**Rule query statement**  
Indicate the source of the messages you want to process with this rule.

**Using SQL version**  
2016-03-23

**Rule query statement**  
SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

```
1 SELECT *, topic(3) as thing FROM '$aws/things/+shadow/update/accepted'
```

---

**Set one or more actions**  
Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (\*.required)

 Send a message as an SNS push notification  
MySNSTopic      [Remove](#)    [Edit](#)

[Add action](#)

---

**Error action**  
Optionally set an action that will be executed when something goes wrong with processing your rule.

[Add action](#)

---

**Tags**  
Apply tags to your resources to help organize and identify them. A tag consists of a case-sensitive key-value pair. [Learn more](#) about tagging your AWS resources.

Tag name	Value	Clear
Provide a tag name, e.g. Manufacturer	Provide a tag value, e.g. Acme-Corporation	<a href="#">Clear</a>

[Add another](#)

---

75

[Cancel](#) [Create rule](#)

To test the rule, add a subscription to the SNS topic you created, and update the shadow of any AWS IoT thing.

You can use the AWS IoT console to find a thing, open its details page, and change the device's shadow. When the Device Shadow service is notified of the change, it publishes 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.

# Using the AWS IoT Device SDKs on a Raspberry Pi

The AWS IoT Device and Mobile SDKs help you to connect your devices to AWS IoT easily and quickly. The AWS IoT Device and Mobile 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.

## Important

Before you start this tutorial, complete the steps in [Getting Started with AWS IoT Core \(p. 5\)](#).

These tutorials provide step-by-step instructions for connecting your Raspberry Pi to the [Message Broker for AWS IoT \(p. 244\)](#) using the AWS IoT Device SDK for Embedded C and the AWS IoT Device SDK for JavaScript. After you complete the steps in these tutorials, you can connect to the AWS IoT platform and run the sample applications included with the AWS IoT Device and Mobile SDKs.

## Contents

- [Prerequisites \(p. 77\)](#)
- [Create an AWS IoT Thing for Your Raspberry Pi \(p. 77\)](#)
- [Using the AWS IoT Device SDK for Embedded C \(p. 78\)](#)
- [Using the AWS IoT Device SDK for JavaScript and Node \(p. 80\)](#)

## Prerequisites

To complete this tutorial, you need the following:

- A [Raspberry Pi 2 Model B](#) or more recent model.
- [Raspbian Wheezy](#) or later. We recommend using the latest version of Raspbian, which is available from the [Raspberry Pi website](#).
- An ethernet or Wi-Fi connection.
- An AWS account. If you don't already have an AWS account, you can get one for free by going to the [Amazon AWS Getting Started Resource Center](#).

## Create an AWS IoT Thing for Your Raspberry Pi

A *thing* represents a device whose state is stored in the AWS Cloud. The device's state is stored in a JSON document known as the device's *shadow*. The shadow is used to both store and retrieve state information. The [Device Shadow service](#) maintains a shadow for each device that is registered in AWS IoT.

### To register your Raspberry Pi with AWS IoT

1. On your Raspberry Pi, browse to the AWS IoT console.
2. In the navigation pane, choose **Secure**, and then choose **Policies**.
3. On the **Policies** page, choose **Create a policy**.

4. On the **Create a policy** page:

- a. Enter a name for the policy (for example, **RaspberryPi-Policy**).
- b. For **Action**, enter **iot:\***.
- c. For **Resource ARN**, enter **\***.
- d. Under **Effect**, choose **Allow**, and then choose **Create**.

This policy allows your Raspberry Pi to publish messages to AWS IoT.

**Important**

These settings are overly permissive. In a production environment narrow the scope of the permissions to that which are required by your device. For more information, see [Authorization \(p. 137\)](#).

5. In the console navigation pane, choose **Manage**, and then choose **Things**.
6. Choose **Create**.
7. On the **Creating AWS IoT things** page, choose **Create a single thing**.
8. On the **Add your device to the device registry** page, enter **RaspberryPi**, and then choose **Next**.
9. On the **Add a certificate for your thing** page, choose **Create certificate**.
10. On the **Certificate created** page, download your private key, device certificate, and a root certificate authority (CA) for AWS IoT. (Choose the **Download** link for each.) These files are saved in your /home/pi/Downloads directory.
11. Choose **Activate** to activate the X.509 certificate, and then choose **Attach a policy**.
12. On the **Add a policy for your thing** page, choose **RaspberryPi-policy** and then choose **Register Thing**.

## Using the AWS IoT Device SDK for Embedded C

This section describes how to run the AWS IoT Device SDK for Embedded C.

### Install the AWS IoT Device SDK for Embedded C

The AWS IoT Device SDK for Embedded C is generally targeted at resource constrained devices that require an optimized C language runtime, but can be used on any operating system and hosted on any processor type (for example, MCUs and MPUs). If you have more memory and processing resources available, you're encouraged to use one of the higher order AWS IoT Device and Mobile SDKs (for example, C++, Java, JavaScript, and Python). In general, the AWS IoT Device SDK for Embedded C is intended for systems that use MCUs or low-end MPUs that run embedded operating systems. For programming examples in the documentation, we use Raspberry Pi running embedded Linux.

#### Example

1. Download the AWS IoT Device SDK for Embedded C to your Raspberry Pi from [GitHub](#).

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

This creates a directory named aws-iot-device-sdk-embedded-c in the current directory.

2. Download mbed TLS to your Raspberry Pi from the [mbed TLS website](#).
3. Navigate into the /home/pi/Downloads directory. Expand the **mbedtls-versionNumber-apache.tgz** file for the latest version by using the following command.

```
tar -xvf mbedtls-versionNumber-apache.tgz
```

4. Copy the contents of `mbedtls-versionNumber` into the `aws-iot-device-sdk-embedded-C/external_libs/mbedtls` directory using the following command.

```
mv ~/Downloads/mbedtls-versionNumber/* ~/aws-iot-device-sdk-embedded-c/external_libs/  
mbedtls
```

## Sample App Configuration

The AWS IoT Device SDK for Embedded C includes sample applications for you to try. For simplicity, this tutorial uses the `subscribe_publish_sample` application, which illustrates how to connect to the AWS IoT Core message broker and subscribe and publish to MQTT topics.

1. Copy the certificate, private key, and root CA certificate you created in [Create an AWS IoT Thing for Your Raspberry Pi \(p. 77\)](#) into the `aws-iot-device-sdk-embedded-C/certs` directory.

**Note**

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

2. You must configure the sample with your personal AWS IoT Core endpoint, private key, certificate, and root CA certificate. Navigate to the `aws-iot-device-sdk-embedded-C/samples/linux/subscribe_publish_sample` directory.

If you have the AWS CLI installed, you can use the `aws iot describe-endpoint --endpoint-type iot:Data-ATS` command to find your personal endpoint URL. If you don't have the AWS CLI installed, open the AWS IoT Core console. From the navigation pane, choose **Manage**, and then choose **Things**. Choose the IoT thing for your Raspberry Pi, and then choose **Interact**. Your endpoint is displayed in the **HTTPS** section of the thing details page.

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   "device.pem.crt"
#define AWS_IOT_PRIVATE_KEY_FILENAME   "private.pem.key"
// =====
```

## Run Sample Applications

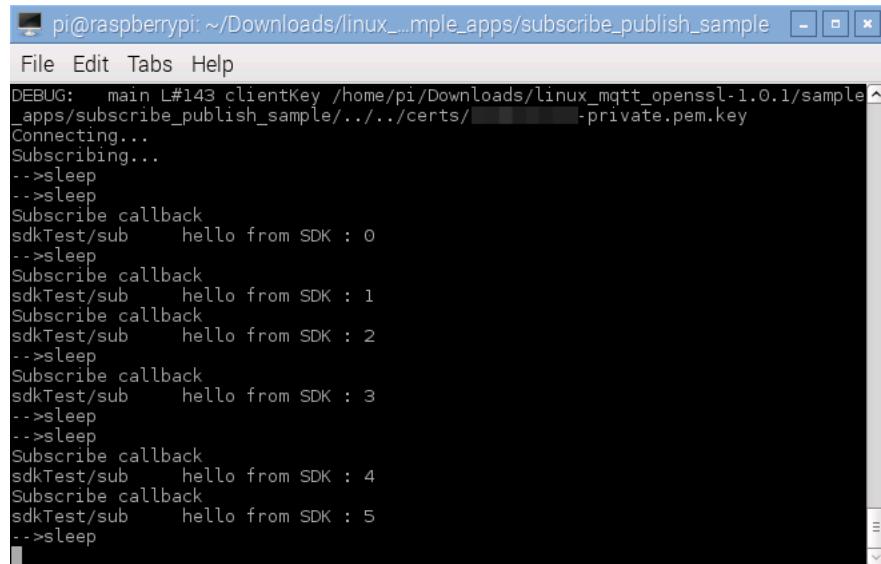
### To run the AWS IoT Device SDK for Embedded C sample applications

1. Use the included makefile to compile the `subscribe_publish_sample_app`.

```
make -f Makefile
```

This generates an executable file.

2. Run the `subscribe_publish_sample_app`. You should see output similar to the following:



```
pi@raspberrypi: ~/Downloads/linux_sample_apps/subscribe_publish_sample
File Edit Tabs Help
DEBUG: main L#143 clientKey /home/pi/Downloads/linux_mqtt_openssl-1.0.1/sample_apps/subscribe_publish_sample/.../certs/-private.pem.key
Connecting...
Subscribing...
-->sleep
-->sleep
Subscribe callback
sdkTest/sub    hello from SDK : 0
-->sleep
Subscribe callback
sdkTest/sub    hello from SDK : 1
Subscribe callback
sdkTest/sub    hello from SDK : 2
-->sleep
Subscribe callback
sdkTest/sub    hello from SDK : 3
-->sleep
Subscribe callback
sdkTest/sub    hello from SDK : 4
Subscribe callback
sdkTest/sub    hello from SDK : 5
-->sleep
```

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

## Using the AWS IoT Device SDK for JavaScript and Node

This tutorial shows you how to install Node.js, the npm package manager, and the AWS IoT Device SDK for JavaScript on a Raspberry Pi and run the sample applications.

### Install the Latest Version of Node.js

To use the AWS IoT Device SDK for JavaScript, install Node.js and the npm package manager on your Raspberry Pi.

1. To download the latest version of the Node repository, open a terminal window and run the following command:

- ```
curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
```
2. Run the following command to install Node and npm.  
**sudo apt-get install -y nodejs**
  3. Run the following commands to verify the installation of Node and npm.  
**node -v**  
and  
**npm -v**  
If a version number is displayed, node and npm are installed correctly.

## Install the AWS IoT Device SDK for JavaScript

Install the AWS IoT Device SDK for JavaScript on your Raspberry Pi.

1. In the terminal window, use the following command to clone the AWS IoT Device SDK for JavaScript repository into your home directory (by default, /home/pi).  
**git clone https://github.com/aws/aws-iot-device-sdk-js.git**
2. **cd** into the `aws-iot-device-sdk-js` directory.
3. Use npm to install the SDK:  
**npm install**

## Prepare to Run the Sample Applications

Under `aws-iot-device-sdk-js`, create a `certs` directory and copy your private key, device certificate, and root CA certificate into it.

To run the AWS IoT Device SDK for JavaScript samples, you need the following information:

### Your AWS Region

To find the Region you are using, open the AWS IoT console and look at the URL. The Region appears immediately after the `https://`. For example:

```
https://us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/dashboard
```

For more information, see [AWS IoT Service Endpoints](#).

### A client ID

An arbitrary alphanumeric string used to identify a device or application that connects to AWS IoT.  
This should be the same as your IoT thing name for this tutorial.

### Your private key

The fully qualified path to your private key on your Raspberry Pi (for example, `/home/pi/aws-iot-device-sdk-js/certs/private.pem.key`).

### Your AWS IoT X.509 certificate

The fully qualified path to your AWS IoT certificate on your Raspberry Pi (for example, `/home/pi/aws-iot-device-sdk-js/certs/device.pem.crt`).

### The STS Amazon root CA

The fully qualified path to the Amazon root CA on your Raspberry Pi (for example, /home/pi/aws-iot-device-sdk-js/certs/Amazon-root-CA-1.pem).

### Your AWS IoT endpoint

If you have installed the AWS CLI on your Raspberry Pi, you can run the **describe-endpoint** CLI command to find your endpoint:

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

If you don't have the AWS CLI installed, open the AWS IoT console. From the navigation pane, choose **Manage**, and then choose **Things**. Choose the IoT thing for your Raspberry Pi, and then choose **Interact**. Your endpoint is displayed in the **HTTPS** section of the thing details page.

### The port on which the AWS IoT message broker listens

This is always 8883.

### Your IoT thing name

This is the name you used when you registered your Raspberry Pi with AWS IoT.

## Run the Sample Applications

The AWS IoT Device SDK for JavaScript contains a number of samples in the aws-iot-device-sdk-js/examples directory. We recommend that you start with device-example.js. This example runs in two modes. In mode 1, it subscribes to the MQTT topic, topic\_1, and publishes a message every 4 seconds on topic\_2. In mode 2, it subscribes to topic\_2 and publishes a message every 4 seconds on topic\_1. You can run two instances of device-example.js (one in mode 1 and one in mode 2) and see the messages being sent and received.

From the aws-iot-device-sdk-js/examples directory, run the following command to start an instance of the sample:

```
node device-example -k "../certs/private.pem.key" -c "../certs/device.pem.crt" -i "raspberry-pi-1" -a "../certs/Amazon-root-CA-1" -H "<your-iot-endpoint>" -p 8883 -T "your-thing-name" --test-mode 1
```

Start another instance of device-example.js running in mode 2:

```
node device-example -k "../certs/private.pem.key" -c "../certs/device.pem.crt" -i "raspberry-pi-2" -a "../certs/Amazon-root-CA-1" -H "<your-iot-endpoint>" -p 8883 -T "your-thing-name" --test-mode 2
```

### Important

Make sure that you use different client IDs when you run the two instances of device-example.js. No two clients (devices or applications) can connect to AWS IoT using the same client ID. The first client's connection is terminated and the second client connection is established.

The thing name is important only when you create a policy specific to an IoT thing. In the AWS IoT Getting Started tutorial, you do not create such a policy, so you can use the same thing name for both instances.

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

If the sample instances are running correctly, the output from the instance running in mode 1 should look like this:

```
substituting 250ms delay for truems...
connect
message topic_1 {"mode1Process":1}
message topic_1 {"mode1Process":2}
message topic_1 {"mode1Process":3}
message topic_1 {"mode1Process":4}
...
```

The output from the instance running in mode 2 should look like this:

```
substituting 250ms delay for truems...
connect
message topic_2 {"mode2Process":1}
message topic_2 {"mode2Process":2}
message topic_2 {"mode2Process":3}
message topic_2 {"mode2Process":4}
...
```

If the sample does not run correctly, try adding the `-d` option to display debug information.

# Other AWS IoT Tutorials

The tutorials in this section show you how to use multiple AWS IoT services together to accomplish a task. These tutorials focus more on the integration of the services rather than a thorough walkthrough of AWS IoT features. The tutorials in this section might build on one another to show how you can start small and evolve your solutions as your business needs change or grow.

Each tutorial includes a list of prerequisites, including specific hardware. Where possible, the tutorials provide alternatives if you don't have all of the required hardware.

## Monitoring Soil Moisture with AWS IoT and Raspberry Pi

This tutorial shows you how to use a [Raspberry Pi](#), a moisture sensor, and AWS IoT to monitor the soil moisture level for a house plant or garden. The Raspberry Pi runs code that reads the moisture level and temperature from the sensor and then sends the data to AWS IoT. You create a rule in AWS IoT that sends an email to an address subscribed to an Amazon SNS topic when the moisture level falls below a threshold.

### Contents

- [Prerequisites \(p. 84\)](#)
- [Setting Up AWS IoT \(p. 84\)](#)
  - [Create the AWS IoT Policy \(p. 85\)](#)
  - [Create the AWS IoT Thing, Certificate, and Private Key \(p. 86\)](#)
  - [Create an Amazon SNS Topic and Subscription \(p. 87\)](#)
  - [Create an AWS IoT Rule to Send an Email \(p. 87\)](#)
- [Setting Up Your Raspberry Pi and Moisture Sensor \(p. 88\)](#)

## Prerequisites

To complete this tutorial, you need:

- An AWS account.
- An IAM user with administrator permissions.
- A development computer running Windows, macOS, Linux, or Unix to access the [AWS IoT console](#).
- A [Raspberry Pi 3B or 4B](#) running [Raspbian Buster](#). For installation instructions, see [Installing operating system images](#) on the Raspberry Pi website.
- A monitor, keyboard, mouse, and Wi-Fi network or Ethernet connection for your Raspberry Pi.
- A Raspberry Pi-compatible moisture sensor. The sensor used in this tutorial is an [Adafruit STEMMA I2C Capacitive Moisture Sensor](#) with a [JST 4-pin to female socket cable header](#).

## Setting Up AWS IoT

To complete this tutorial, you need to create the following resources. To connect a device to AWS IoT, you create an IoT thing, a device certificate, and an AWS IoT policy.

- An AWS IoT thing.

A thing represents a physical device (in this case, your Raspberry Pi) and contains static metadata about the device.

- A device certificate.

All devices must have a device certificate to connect to and authenticate with AWS IoT.

- An AWS IoT policy.

Each device certificate has one or more AWS IoT policies associated with it. These policies determine which AWS IoT resources the device can access.

- An AWS IoT root CA certificate.

Devices and other clients use an AWS IoT root CA certificate to authenticate the AWS IoT server with which they are communicating. For more information, see [Server Authentication](#).

- An AWS IoT rule.

A rule contains a query and one or more rule actions. The query extracts data from device messages to determine if the message data should be processed. The rule action specifies what to do if the data matches the query.

- An Amazon SNS topic and topic subscription.

The rule listens for moisture data from your Raspberry Pi. If the value is below a threshold, it sends a message to the Amazon SNS topic. Amazon SNS sends that message to all email addresses subscribed to the topic.

## Create the AWS IoT Policy

Create an AWS IoT policy that allows your Raspberry Pi to connect and send messages to AWS IoT.

1. In the [AWS IoT console](#), if a **Get started** button appears, choose it. Otherwise, in the navigation pane, expand **Secure**, and then choose **Policies**.
2. If a **You don't have any policies yet** dialog box appears, choose **Create a policy**. Otherwise, choose **Create**.
3. Enter a name for the AWS IoT policy (for example, **MoistureSensorPolicy**).
4. In the **Add statements** section, replace the existing policy with the following JSON. Replace **<region>** and **<account>** with your AWS Region and AWS account number.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:Connect",
            "Resource": "arn:aws:iot:<region>:<account>:client/RaspberryPi"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Publish",
            "Resource": [
                "arn:aws:iot:<region>:<account>:topic/$aws/things/RaspberryPi/shadow/update",
                "arn:aws:iot:<region>:<account>:topic/$aws/things/RaspberryPi/shadow/delete",
                "arn:aws:iot:<region>:<account>:topic/$aws/things/RaspberryPi/shadow/get"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Receive",
            "Resource": "arn:aws:iot:<region>:<account>:topic/$aws/things/RaspberryPi"
        }
    ]
}
```

```

        "Resource": [
            "arn:aws:iot:<region>:<account>:topic/$aws/things/RaspberryPi/shadow/
update/accepted",
            "arn:aws:iot:<region>:<account>:topic/$aws/things/RaspberryPi/shadow/
delete/accepted",
            "arn:aws:iot:<region>:<account>:topic/$aws/things/RaspberryPi/shadow/get/
accepted",
            "arn:aws:iot:<region>:<account>:topic/$aws/things/RaspberryPi/shadow/
update/rejected",
            "arn:aws:iot:<region>:<account>:topic/$aws/things/RaspberryPi/shadow/
delete/rejected"
        ],
    },
    {
        "Effect": "Allow",
        "Action": "iot:Subscribe",
        "Resource": [
            "arn:aws:iot:<region>:<account>:topicfilter/$aws/things/RaspberryPi/shadow/
update/accepted",
            "arn:aws:iot:<region>:<account>:topicfilter/$aws/things/RaspberryPi/shadow/
delete/accepted",
            "arn:aws:iot:<region>:<account>:topicfilter/$aws/things/RaspberryPi/shadow/
get/accepted",
            "arn:aws:iot:<region>:<account>:topicfilter/$aws/things/RaspberryPi/shadow/
update/rejected",
            "arn:aws:iot:<region>:<account>:topicfilter/$aws/things/RaspberryPi/shadow/
delete/rejected"
        ],
    },
    {
        "Effect": "Allow",
        "Action": [
            "iot:GetThingShadow",
            "iot:UpdateThingShadow",
            "iot:DeleteThingShadow"
        ],
        "Resource": "arn:aws:iot:<region>:<account>:thing/RaspberryPi"
    }
]
}

```

5. Choose **Create**.

## Create the AWS IoT Thing, Certificate, and Private Key

Create a thing in the AWS IoT registry to represent your Raspberry Pi.

1. In the [AWS IoT console](#), in the navigation pane, choose **Manage**, and then choose **Things**.
2. If a **You don't have any things yet** dialog box is displayed, choose **Register a thing**. Otherwise, choose **Create**.
3. On the **Creating AWS IoT things** page, choose **Create a single thing**.
4. On the **Add your device to the device registry** page, enter a name for your IoT thing (for example, **RaspberryPi**), and then choose **Next**. You can't change the name of a thing after you create it. To change a thing's name, you must create a new thing, give it the new name, and then delete the old thing.
5. On the **Add a certificate for your thing** page, choose **Create certificate**.
6. Choose the **Download** links to download the certificate, private key, and root CA certificate.

### Important

This is the only time you can download your certificate and private key.

7. Choose **Activate**.
8. Choose **Attach a policy**.
9. For **Add a policy for your thing**, choose **MoistureSensorPolicy**, and then choose **Register Thing**.

## Create an Amazon SNS Topic and Subscription

Create an Amazon SNS topic and subscription.

1. From the AWS IoT console, choose **Services**, type **SNS**, and then choose **Simple Notification Service**.
2. In the navigation pane, choose **Topics**, and then choose **Create topic**.
3. Enter a name for the topic (for example, **MoistureSensorTopic**).
4. Enter a display name for the topic (for example, **Moisture Sensor Topic**). This is the name displayed for your topic in the Amazon SNS console.
5. Choose **Create topic**.
6. In the Amazon SNS topic detail page, choose **Create subscription**.
7. For **Protocol**, choose **Email**.
8. For **Endpoint**, enter your email address.
9. Choose **Create subscription**.
10. Open your email client and look for a message with the subject **MoistureSensorTopic**. Open the email and click the **Confirm subscription** link.

### Important

You won't receive any email alerts from this Amazon SNS topic until you confirm the subscription.

You should receive an email message with the text you typed.

## Create an AWS IoT Rule to Send an Email

An AWS IoT rule defines a query and one or more actions to take when a message is received from a device. The AWS IoT rules engine listens for messages sent by devices and uses the data in the messages to determine if some action should be taken. For more information, see [Rules for AWS IoT \(p. 260\)](#).

In this tutorial, your Raspberry Pi publishes messages on `aws/things/RaspberryPi/shadow/update`. This is an internal MQTT topic used by devices and the Thing Shadow service. The Raspberry Pi publishes messages that have the following form:

```
{  
  "reported": {  
    "moisture" : <moisture-reading>,  
    "temp" : <temperature-reading>  
  }  
}
```

You create a query that extracts the moisture and temperature data from the incoming message. You also create an Amazon SNS action that takes the data and sends it to Amazon SNS topic subscribers if the moisture reading is below a threshold value.

### Create an Amazon SNS rule

1. In the [AWS IoT console](#), in the navigation pane, choose **Act**. If a **You don't have any rules yet** dialog box appears, choose **Create a rule**. Otherwise, choose **Create**.

2. In the **Create a rule** page, enter a name for your rule (for example, **MoistureSensorRule**).
3. For **Description**, provide a short description for this rule (for example, **Sends an alert when soil moisture level readings are too low**).
4. Under **Rule query statement**, choose SQL version **2016-03-23**, and enter the following AWS IoT SQL query statement:

```
SELECT * FROM '$aws/things/RaspberryPi/shadow/update/accepted' WHERE state.reported.moisture < 400
```

This statement triggers the rule action when the `moisture` reading is less than 400.

**Note**

You might have to use a different value. After you have the code running on your Raspberry Pi, you can see the values you get from your sensor by touching the sensor, placing it in water, or placing it in a planter.

5. Under **Set one or more actions**, choose **Add action**.
6. On the **Select an action** page, choose **Send a message as an SNS push notification**.
7. Scroll to the bottom of the page, and then choose **Configure action**.
8. On the **Configure action** page, for **SNS target**, choose **Select**, and then choose **LowMoistureTopic**.
9. For **Message format**, choose **RAW**.
10. Under **Choose or create a role to grant AWS IoT access to perform this action**, choose **Create role**. Enter a name for the role (for example, **LowMoistureTopicRole**), and then choose **Create role**.
11. Choose **Add action**.
12. Choose **Create rule**.

## Setting Up Your Raspberry Pi and Moisture Sensor

Insert your micro SD card into the Raspberry Pi, connect your monitor, keyboard, mouse, and, if you're not using Wi-Fi, Ethernet cable. Do not connect the power cable yet.

Connect the JST jumper cable to the moisture sensor. The other side of the jumper has four wires:

- Green: I2C SCL
- White: I2C SDA
- Red: power (3.5 V)
- Black: ground

Hold the Raspberry Pi with the Ethernet jack on the right. In this orientation, there are two rows of GPIO pins at the top. Connect the wires from the moisture sensor to the bottom row of pins in the following order. Starting at the left-most pin, connect red (power), white (SDA), and green (SCL). Skip one pin, and then connect the black (ground) wire. For more information, see [Python Computer Wiring](#).

Attach the power cable to the Raspberry Pi and plug the other end into a wall socket to turn it on.

### Configure your Raspberry Pi

1. On **Welcome to Raspberry Pi**, choose **Next**.
2. Choose your country, language, timezone, and keyboard layout. Choose **Next**.
3. Enter a password for your Raspberry Pi, and then choose **Next**.

4. Choose your Wi-Fi network, and then choose **Next**. If you aren't using a Wi-Fi network, choose **Skip**.
5. Choose **Next** to check for software updates. When the updates are complete, choose **Restart** to restart your Raspberry Pi.

After your Raspberry Pi starts up, enable the I2C interface.

1. In the upper left corner of the Raspbian desktop, click the Raspberry icon, choose **Preferences**, and then choose **Raspberry Pi Configuration**.
2. On the **Interfaces** tab, for **I2C**, choose **Enable**.
3. Choose **OK**.

The libraries for the Adafruit STEMMA moisture sensor are written for CircuitPython. To run them on a Raspberry Pi, you need to install the latest version of Python 3.

1. Run the following commands from a command prompt to update your Raspberry Pi software:

```
sudo apt-get update  
sudo apt-get upgrade
```

2. Run the following command to update your Python 3 installation:

```
sudo pip3 install --upgrade setuptools
```

3. Run the following command to install the Raspberry Pi GPIO libraries:

```
pip3 install RPI.GPIO
```

4. Run the following command to install the Adafruit Blinka libraries:

```
pip3 install adafruit-blinka
```

For more information, see [Installing CircuitPython Libraries on Raspberry Pi](#).

5. Run the following command to install the Adafruit Seesaw libraries:

```
sudo pip3 install adafruit-circuitpython-seesaw
```

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

```
pip3 install AWSIoTPythonSDK
```

Your Raspberry Pi now has all of the required libraries. Create a file called **moistureSensor.py** and copy the following Python code into the file:

```
from adafruit_seesaw.seesaw import Seesaw  
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTShadowClient  
from board import SCL, SDA  
  
import logging  
import time  
import json  
import argparse  
import busio  
  
# Shadow JSON schema:  
#  
# {  
#     "state": {  
#         "desired":{  
#             "moisture":<INT VALUE>,
```

```

#           "temp":<INT VALUE>
#       }
#   }
# }

# Function called when a shadow is updated
def customShadowCallback_Update(payload, responseStatus, token):

    # Display status and data from update request
    if responseStatus == "timeout":
        print("Update request " + token + " time out!")

    if responseStatus == "accepted":
        payloadDict = json.loads(payload)
        print("~~~~~")
        print("Update request with token: " + token + " accepted!")
        print("moisture: " + str(payloadDict["state"]["reported"]["moisture"]))
        print("temperature: " + str(payloadDict["state"]["reported"]["temp"]))
        print("~~~~~\n\n")

    if responseStatus == "rejected":
        print("Update request " + token + " rejected!")

# Function called when a shadow is deleted
def customShadowCallback_Delete(payload, responseStatus, token):

    # Display status and data from delete request
    if responseStatus == "timeout":
        print("Delete request " + token + " time out!")

    if responseStatus == "accepted":
        print("~~~~~")
        print("Delete request with token: " + token + " accepted!")
        print("~~~~~\n\n")

    if responseStatus == "rejected":
        print("Delete request " + token + " rejected!")

# Read in command-line parameters
def parseArgs():

    parser = argparse.ArgumentParser()
    parser.add_argument("-e", "--endpoint", action="store", required=True, dest="host",
    help="Your AWS IoT custom endpoint")
    parser.add_argument("-r", "--rootCA", action="store", required=True, dest="rootCAPath",
    help="Root CA file path")
    parser.add_argument("-c", "--cert", action="store", dest="certificatePath",
    help="Certificate file path")
    parser.add_argument("-k", "--key", action="store", dest="privateKeyPath", help="Private
    key file path")
    parser.add_argument("-p", "--port", action="store", dest="port", type=int, help="Port
    number override")
    parser.add_argument("-n", "--thingName", action="store", dest="thingName",
    default="Bot", help="Targeted thing name")
    parser.add_argument("-id", "--clientId", action="store", dest="clientId",
    default="basicShadowUpdater", help="Targeted client id")

    args = parser.parse_args()
    return args

# Configure logging
# AWSIoTMQTTShadowClient writes data to the log
def configureLogging():

```

```
logger = logging.getLogger("AWSIoTPythonSDK.core")
logger.setLevel(logging.DEBUG)
streamHandler = logging.StreamHandler()
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
streamHandler.setFormatter(formatter)
logger.addHandler(streamHandler)

# Parse command line arguments
args = parseArgs()

if not args.certificatePath or not args.privateKeyPath:
    parser.error("Missing credentials for authentication.")
    exit(2)

# If no --port argument is passed, default to 8883
if not args.port:
    args.port = 8883

# Init AWSIoTMQTTShadowClient
myAWSIoTMQTTShadowClient = None
myAWSIoTMQTTShadowClient = AWSIoTMQTTShadowClient(args.clientId)
myAWSIoTMQTTShadowClient.configureEndpoint(args.host, args.port)
myAWSIoTMQTTShadowClient.configureCredentials(args.rootCAPath, args.privateKeyPath,
  args.certificatePath)

# AWSIoTMQTTShadowClient connection configuration
myAWSIoTMQTTShadowClient.configureAutoReconnectBackoffTime(1, 32, 20)
myAWSIoTMQTTShadowClient.configureConnectDisconnectTimeout(10) # 10 sec
myAWSIoTMQTTShadowClient.configureMQTTOperationTimeout(5) # 5 sec

# Initialize Raspberry Pi's I2C interface
i2c_bus = busio.I2C(SCL, SDA)

# Intialize SeeSaw, Adafruit's Circuit Python library
ss = Seesaw(i2c_bus, addr=0x36)

# Connect to AWS IoT
myAWSIoTMQTTShadowClient.connect()

# Create a device shadow handler, use this to update and delete shadow document
deviceShadowHandler = myAWSIoTMQTTShadowClient.createShadowHandlerWithName(args.thingName,
                           True)

# Delete current shadow JSON doc
deviceShadowHandler.shadowDelete(customShadowCallback_Delete, 5)

# Read data from moisture sensor and update shadow
while True:

    # read moisture level through capacitive touch pad
    moistureLevel = ss.moisture_read()

    # read temperature from the temperature sensor
    temp = ss.get_temp()

    # Display moisture and temp readings
    print("Moisture Level: {}".format(moistureLevel))
    print("Temperature: {}".format(temp))

    # Create message payload
    payload = {"state":{"reported":{"moisture":str(moistureLevel), "temp":str(temp)}}}

    # Update shadow
    deviceShadowHandler.shadowUpdate(json.dumps(payload), customShadowCallback_Update, 5)
```

```
time.sleep(1)
```

Save the file to a place you can find it. Run `moistureSensor.py` from the command line with the following parameters:

`endpoint`

Your custom AWS IoT endpoint. For more information, see [Device Shadow RESTful API \(p. 367\)](#).

`rootCA`

The full path to your AWS IoT root CA certificate.

`cert`

The full path to your AWS IoT device certificate.

`key`

The full path to your AWS IoT device certificate private key.

`thingName`

Your thing name (in this case, `RaspberryPi`).

`clientId`

The MQTT client ID. Use `RaspberryPi`.

The command line should look like this:

```
python3 moistureSensor.py --endpoint <your-endpoint> --rootCA ~/certs/  
AmazonRootCA1.pem --cert ~/certs/raspberrypi-certificate.pem.crt --key ~/certs/  
raspberrypi-private.pem.key --thingName RaspberryPi --clientId RaspberryPi
```

Try touching the sensor, putting it in a planter, or putting it in a glass of water to see how the sensor responds to various levels of moisture. If needed, you can change the threshold value in the `MoistureSensorRule`. When the moisture sensor reading goes below the value specified in your rule's SQL query statement, AWS IoT publishes a message to the Amazon SNS topic. You should receive an email message that contains the moisture and temperature data.

After you have verified receipt of email messages from Amazon SNS, press **CTRL+C** to stop the Python program. It is unlikely the Python program will send enough messages to incur charges, but it is a best practice to stop the program when you are done.

# Managing Devices with AWS IoT

AWS IoT provides a registry that helps you manage *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 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 registry and the Device Shadow service. This provides organization and convenience to your IoT fleet without removing the flexibility of the underlying device certificate model or shadows.

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

## How to Manage Things with the 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 registry.

### Create a Thing

The following command shows how to use the AWS IoT **CreateThing** command from the CLI to create a thing. You can't change a thing's name after you create it. To change a thing's name, you must create a new thing, give it the new name, and then delete the old thing.

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

The **CreateThing** command displays the name and Amazon Resource Name (ARN) of your new thing:

```
{  
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",  
    "thingName": "MyLightBulb"  
    "thingId": "12345678abcdefghijklmnopqrstuvwxyz"  
}
```

**Note**

We do not recommend using personally identifiable information in your thing names.

## 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": "12345678abcdefghijklmnpqrstuvwxyz"
    "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. Note that this command updates only the thing's attributes. You can't change a thing's name. To change a thing's name, you must create a new thing, give it the new name, and then delete the old 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"
```

This command returns successfully with no error if the deletion is successful or you specify a thing that doesn't exist.

## Attach a Principal to a Thing

A physical device must have an X.509 certificate to communicate with AWS IoT. You can associate the certificate on your device with the thing in the 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 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 makes it easier to discover 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 be associated with only 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",  
    "thingTypeId": "df9c2d8c-894d-46a9-8192-9068d01b2886",  
    "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 returns information about the specified type:

```
{  
    "thingTypeProperties": {  
        "searchableAttributes": [  
            "model",  
            "wattage"  
        ],  
        "thingTypeDescription": "light bulb type"  
},  
    "thingTypeId": "df9c2d8c-894d-46a9-8192-9068d01b2886",  
    "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb",  
    "thingTypeName": "LightBulb",  
    "thingTypeMetadata": {  

```

## 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 "MyLightBulb" --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 "LightBulb" --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 are 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",  
            "color",  
            "brightness"  
        ]  
    },  
    "thingTypeDescription": "A stop light thing type."  
}
```

```
        "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": "12345678abcdefg12345678ijklmnop12345678"
    "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.

## Static Thing Groups

Static thing groups allow you to manage several things at once by categorizing them into groups. Static thing groups contain a group of things that are managed by using the console, CLI, or the API. [Dynamic thing groups](#), on the other hand, contain things that match a specified query. Static thing groups can also contain other static thing groups — you can build a hierarchy of groups. You can attach a policy to a parent group and it is inherited by its child groups, and by all of the things in the group and in its child groups. This makes control of permissions easy for large numbers of things.

Here are the things you can do with static 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 \(p. 211\)](#).
- Create jobs that are sent to and executed on every thing in a group and its child groups. See [Jobs \(p. 381\)](#).

Here are some limitations of static thing groups:

- A group can have at most one direct parent.
- If a group is a child of another group, you must specify this 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 you create any child groups it contains.
- The number of groups to which a thing can belong is [limited](#).
- 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 that share a common parent.)
- You cannot rename a group.
- Thing group names can't contain international characters, such as û, é and ñ.

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 Static Thing Group

Use the **CreateThingGroup** command to create a static thing group:

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

The **CreateThingGroup** command returns a response that contains the static thing group's name, ID, and ARN:

```
{  
    "thingGroupName": "LightBulbs",
```

```
        "thingGroupId": "abcdefghijklmnopqrstuvwxyz12345678ijklmnop12345678qrstuvwxyz",
        "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
    }
```

**Note**

We do not recommend using personally identifiable information in your thing group names.

Here is an example that specifies a parent of the static 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 static thing group's name,, ID, and ARN:

```
{
    "thingGroupName": "RedLights",
    "thingGroupId": "abcdefghijklmnopqrstuvwxyz12345678ijklmnop12345678qrstuvwxyz",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
}
```

**Important**

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

- A thing group can have only one direct parent.
- The number of direct child groups a thing group can have is [limited](#).
- The maximum depth of a group hierarchy is [limited](#).
- The number of attributes a thing group can have is [limited](#). (Attributes are name-value pairs you can use to store information about a group.) The lengths of each attribute name and each value are also [limited](#).

## 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 returns information about the specified group:

```
{
    "thingGroupName": "RedLights",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
    "thingGroupId": "12345678abcdefghijklmnopqrstuvwxyz12345678ijklmnop12345678",
    "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 a Thing to a Static Thing Group

You can use the **AddThingToThingGroup** command to add a thing to a static thing 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.)

If a thing belongs to as many thing groups as possible, and one or more of those groups is a dynamic thing group, you can use the [overrideDynamicGroups](#) flag to make static groups take priority over dynamic groups.

## Remove a Thing from a Static 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 a Thing Group

You can use the **ListThingsInThingGroup** command to list the things that belong 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:

```
$ 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 might not be reflected immediately.

## List Thing Groups

You can use the **ListThingGroups** command to list your account's thing groups:

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

The **ListThingGroups** command returns a list of the thing groups 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, 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 might not be reflected immediately.

## List Groups for a 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 Static Thing Group

You can use the **UpdateThingGroup** command to update the attributes of a static 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

The number of attributes that a thing can have is [limited](#).

## 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 that has child thing groups, you receive 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 you delete the group.

You can delete a group that has child things, but any permissions granted to the things by membership in the group 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](#)) might continue to show the group while records in the cloud are being updated.

## Attach a Policy to a Static Thing Group

You can use the **AttachPolicy** command to attach a policy to a static 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**

You can attach a maximum number of two policies to a group.

**Note**

We do not recommend using personally identifiable information in your policy names.

The `--target` parameter can be a thing group ARN (as above), a certificate ARN, or an Amazon Cognito Identity. For more information about policies, certificates and authentication, see [Authentication \(p. 120\)](#).

## Detach a Policy from a Static 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 "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"  
--policy-name "myLightBulbPolicy"
```

The **DetachPolicy** command does not produce any output.

## List the Policies Attached to a Static Thing Group

You can use the **ListAttachedPolicies** command to list the policies attached to a static thing group:

```
$ aws iot list-attached-policies --target "arn:aws:iot:us-west-2:123456789012:thinggroup/  
RedLights"
```

The `--target` parameter can be a thing group ARN (as above), a certificate ARN, or an Amazon Cognito identity.

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

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 an Amazon Cognito identity. If you specify only the **--cognito-identity-pool-id**, 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 can 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) are 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/my/topic\"]}"
```

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 Core Policies \(p. 139\)](#) 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 that were allowed or denied for each set of `--auth-infos` queries you specified:

```
{
    "authResults": [
        {
            "allowed": {
                "policies": [
                    {
                        "policyArn": "string",
                        "policyName": "string"
                    }
                ]
            },
            "authDecision": "string",
            "authInfo": {
                "actionType": "string",
                "resources": [ "string" ]
            },
            "denied": {
                "explicitDeny": {
                    "policies": [
                        {
                            "policyArn": "string",
                            "policyName": "string"
                        }
                    ]
                },
                "implicitDeny": {
                    "policies": [
                        {
                            "policyArn": "string",
                            "policyName": "string"
                        }
                    ]
                }
            },
            "missingContextValues": [ "string" ]
        }
    ]
}
```

## Dynamic Thing Groups

Dynamic thing groups update group membership through search queries. Using dynamic thing groups, you can change the way you interact with things depending on their connectivity, registry, or shadow data.

Because dynamic thing groups are tied to your fleet index, you must enable the fleet indexing service to use them. You can preview the things in a dynamic thing group before you create the group with a fleet indexing search query. For more information, see [Fleet Indexing Service](#) and [Fleet Indexing Service: Query Syntax](#).

You can specify a dynamic thing group as a target for a job. Only things that meet the criteria that define the dynamic thing group perform the job.

For example, suppose that you want to update the firmware on your devices, but, to minimize the chance that the update is interrupted, you only want to update firmware on devices with battery life greater than 80%. You can create a dynamic thing group that only includes devices with a reported battery life above 80%, and you can use that dynamic thing group as the target for your firmware update job. Only devices that meet your battery life criteria receive the firmware update. As devices reach the 80% battery life criteria, they are added to the dynamic thing group and receive the firmware update.

For more information about specifying thing groups as job targets, see [Using the AWS IoT Jobs APIs](#).

Dynamic thing groups differ from static thing groups in the following ways:

- Thing membership is not explicitly defined. To create a dynamic thing group, you must define a query string that defines group membership.
- Dynamic thing groups cannot be part of a hierarchy.
- Dynamic thing groups cannot have policies applied to them.
- You use a different set of commands to create, update, and delete dynamic thing groups. For all other operations, the same commands that you use to interact with static thing groups can be used to interact with dynamic thing groups.
- The number of dynamic groups that a single account can have is [limited](#).

For more information about static thing groups, see [Static Thing Groups \(p. 99\)](#).

As an example, suppose we create a dynamic group that contains all rooms in a warehouse whose temperature is greater than 60 degrees Fahrenheit. When a room's temperature is 61 degrees or higher, it is added to the RoomTooWarm dynamic thing group. All rooms in the RoomTooWarm dynamic thing group have cooling fans turned on. When a room's temperature falls to 60 degrees or lower, it is removed from the dynamic thing group and its fan would be turned off.

## Create a Dynamic Thing Group

Use the **CreateDynamicThingGroup** command to create a dynamic thing group. To create a dynamic thing group for the room too warm scenario you would use the **create-dynamic-thing-group** CLI command:

```
$ aws iot create-dynamic-thing-group --thing-group-name "RoomTooWarm" --query-string "attributes.temperature>60"
```

### Note

We do not recommend using personally identifiable information in your dynamic thing group names.

The **CreateDynamicThingGroup** command returns a response that contains the index name, query string, query version, thing group name, thing group ID, and thing group ARN:

```
{  
    "indexName": "AWS_Things",  
    "queryVersion": "2017-09-30",  
    "thingGroupName": "RoomTooWarm",  
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RoomTooWarm",  
    "queryString": "attributes.temperature>60\n",  
    "thingGroupId": "abcdefghijklmnopqrstuvwxyz123456789012345678qrstuvwxyz"  
}
```

Dynamic thing group creation is not instantaneous. The dynamic thing group backfill takes time to complete. When a dynamic thing group is created, the status of the group is set to **BUILDING**. When the backfill is complete, the status changes to **ACTIVE**. To check the status of your dynamic thing group, use the [DescribeThingGroup](#) command.

## Describe a Dynamic Thing Group

Use the **DescribeThingGroup** command to get information about a dynamic thing group:

```
$ aws iot describe-thing-group --thing-group-name "RoomTooWarm"
```

The **DescribeThingGroup** command returns information about the specified group:

```
{  
    "status": "ACTIVE",  
    "indexName": "AWS_Things",  
    "thingGroupName": "RoomTooWarm",  
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RoomTooWarm",  
    "queryString": "attributes.temperature>60\n",  
    "version": 1,  
    "thingGroupMetadata": {  
        "creationDate": 1548716921.289  
    },  
    "thingGroupProperties": {},  
    "queryVersion": "2017-09-30",  
    "thingGroupId": "84dd9b5b-2b98-4c65-84e4-be0e1ecf4fd8"  
}
```

Running **DescribeThingGroup** on a dynamic thing group returns attributes that are specific to dynamic thing groups, such as the `queryString` and the `status`.

The status of a dynamic thing group can take the following values:

**ACTIVE**

The dynamic thing group is ready for use.

**BUILDING**

The dynamic thing group is being created, and thing membership is being processed.

**REBUILDING**

The dynamic thing group's membership is being updated, following the adjustment of the group's search query.

### Note

After you create a dynamic thing group, you can use the group, regardless of its status. Only dynamic thing groups with an **ACTIVE** status include all of the things that match the search query for that dynamic thing group. Dynamic thing groups with **BUILDING** and **REBUILDING** statuses might not include all of the things that match the search query.

## Update a Dynamic Thing Group

Use the **UpdateDynamicThingGroup** command to update the attributes of a dynamic thing group, including the group's search query. The following command updates the thing group description and the query string changing the membership criteria to temperature > 65:

```
$ aws iot update-dynamic-thing-group --thing-group-name "RoomTooWarm" --thing-group-properties "thingGroupDescription=\"This thing group contains rooms warmer than 65F.\" --query-string "attributes.temperature>65"
```

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

```
{  
    "version": 2  
}
```

Dynamic thing group updates are not instantaneous. The dynamic thing group backfill takes time to complete. When a dynamic thing group is updated, the status of the group changes to REBUILDING while the group updates its membership. When the backfill is complete, the status changes to ACTIVE. To check the status of your dynamic thing group, use the **DescribeThingGroup** command.

## Delete a Dynamic Thing Group

Use the **DeleteDynamicThingGroup** command to delete a dynamic thing group:

```
$ aws iot delete-dynamic-thing-group --thing-group-name "RoomTooWarm"
```

The **DeleteDynamicThingGroup** command does not produce any output.

Commands that show which groups a thing belongs to (for example, **ListGroupsForThing**) might continue to show the group while records in the cloud are being updated.

## Limitations and Conflicts

Dynamic thing groups share these limitations with static thing groups:

- The number of attributes a thing group can have is [limited](#).
- The number of groups to which a thing can belong is [limited](#).
- Thing groups cannot be renamed.
- Thing group names cannot contain international characters, such as û, é, and ñ.

When using dynamic thing groups, keep the following in mind.

### The fleet indexing service must be enabled

The fleet indexing service must be enabled and the fleet indexing backfill must be complete before you can create and use dynamic thing groups. Expect a delay after you enable the fleet indexing service. The backfill can take some time to complete. The more things that you have registered, the longer the backfill process takes. After you enable the fleet indexing service for dynamic thing groups, you cannot disable it until you delete all of your dynamic thing groups.

#### Note

If you have permissions to query the fleet index, you can access the data of things across the entire fleet.

### The number of dynamic thing groups is limited

The number of dynamic groups is [limited](#).

## Successful commands can log errors

When creating or updating a dynamic thing group, it's possible that some things might be eligible to be in a dynamic thing group yet not be added to it. The command to create or update a dynamic thing group, however, still succeeds in those cases while logging an error and generating an [AddThingToDynamicThingGroupsFailed metric](#).

An [error log entry](#) in the CloudWatch log is created for each thing when an eligible thing cannot be added to a dynamic thing group or a thing is removed from a dynamic thing group to add it to another group. When a thing cannot be added to a dynamic group, an [AddThingToDynamicThingGroupsFailed metric](#) is also created; however, a single metric can represent multiple log entries.

When a thing becomes eligible to be added to a dynamic thing group, the following is considered:

- Is the thing already in as many groups as it can be? (See [limits](#))
  - **NO:** The thing is added to the dynamic thing group.
  - **YES:** Is the thing a member of any dynamic thing groups?
    - **NO:** The thing can't be added to the dynamic thing group, an error is logged, and an [AddThingToDynamicThingGroupsFailed metric](#) is generated.
    - **YES:** Is the dynamic thing group to join older than any dynamic thing group that the thing is already a member of?
      - **NO:** The thing can't be added to the dynamic thing group, an error is logged, and an [AddThingToDynamicThingGroupsFailed metric](#) is generated.
      - **YES:** Remove the thing from the most recent dynamic thing group it is a member of, log an error, and add the thing to the dynamic thing group. This generates an error and an [AddThingToDynamicThingGroupsFailed metric](#) for the dynamic thing group from which the thing was removed.

When a thing in a dynamic thing group no longer meets the search query, it is removed from the dynamic thing group. Likewise, when a thing is updated to meet a dynamic thing group's search query, it is then added to the group as previously described. These additions and removals are normal and do not produce error log entries.

## With `overrideDynamicGroups` enabled, static groups take priority over dynamic groups

The number of groups to which a thing can belong is [limited](#). When you update thing membership by using the [AddThingToThingGroup](#) or [UpdateThingGroupsForThing](#) commands, adding the `--overrideDynamicGroups` parameter gives static thing groups priority over dynamic thing groups.

When adding a thing to a static thing group, the following is considered:

- Does the thing already belong to the maximum number of groups?
  - **NO:** The thing is added to the static thing group.
  - **YES:** Is the thing in any dynamic groups?
    - **NO:** The thing cannot be added to the thing group. The command raises an exception.
    - **YES:** Was `--overrideDynamicGroups` enabled?
      - **NO:** The thing cannot be added to the thing group. The command raises an exception.
      - **YES:** The thing is removed from the most recently created dynamic thing group, an error is logged, and an [AddThingToDynamicThingGroupsFailed metric](#) is generated for the dynamic thing group from which the thing was removed. Then, the thing is added to the static thing group.

## Older dynamic thing groups take priority over newer ones

The number of groups to which a thing can belong is [limited](#). When a thing becomes eligible to be added to a dynamic thing group because of a create or update operation, and the thing is already in as many groups as it can be, it can be removed from another dynamic thing group to enable this addition. For more information about how this occurs, see [Successful commands can log errors \(p. 112\)](#) and [With overrideDynamicGroups enabled, static groups take priority over dynamic groups \(p. 112\)](#) for examples.

When a thing is removed from a dynamic thing group, an error is logged, and an event is raised.

## You cannot apply policies to dynamic thing groups

Attempting to apply a policy to a dynamic thing group generates an exception.

## Dynamic thing group membership is eventually consistent

Only the final state of a thing is evaluated for the registry. Intermediary states can be skipped if states are updated rapidly. Avoid associating a rule or job, with a dynamic thing group whose membership depends on an intermediary state.

# Tagging Your AWS IoT Resources

To help you manage and organize your thing groups, thing types, topic rules, jobs, scheduled audits and security profiles you can optionally assign your own metadata to each of these resources in the form of tags. This section describes tags and shows you how to create them.

To help you manage your costs related to things, you can create [billing groups \(p. 116\)](#) that contain things. You can then assign tags that contain your metadata to each of these billing groups. This section also discusses billing groups and the commands available to create and manage them.

## Tag Basics

You can use tags to categorize your AWS IoT resources in different ways (for example, by purpose, owner, or environment). This is useful when you have many resources of the same type — you can quickly identify a resource based on the tags you've assigned to it. Each tag consists of a key and optional value, both of which you define. For example, you can define a set of tags for your thing types that helps you track devices by type. We recommend that you create a set of tag keys that meets your needs for each kind of resource. Using a consistent set of tag keys makes it easier for you to manage your resources.

You can search for and filter resources based on the tags you add or apply. You can also use billing group tags to categorize and track your costs. You can also use tags to control access to your resources as described in [Using Tags with IAM Policies \(p. 115\)](#).

For ease of use, the Tag Editor in the AWS Management Console provides a central, unified way to create and manage your tags. For more information, see [Working with Tag Editor](#) in [Working with the AWS Management Console](#).

You can also work with tags using the AWS CLI and the AWS IoT API. You can associate tags with thing groups, thing types, topic rules, jobs, security profiles, and billing groups when you create them by using the `Tags` field in the following commands:

- [CreateBillingGroup](#)
- [CreateDynamicThingGroup](#)
- [CreateJob](#)
- [CreateOTAUpdate](#)
- [CreateScheduledAudit](#)
- [CreateSecurityProfile](#)
- [CreateStream](#)
- [CreateThingGroup](#)
- [CreateThingType](#)
- [CreateTopicRule](#)

You can add, modify, or delete tags for existing resources that support tagging by using the following commands:

- [TagResource](#)
- [ListTagsForResource](#)

- [UntagResource](#)

You can edit tag keys and values, and you can remove tags from a resource at any time. You can set the value of a tag to an empty string, but you can't set the value of a tag to null. If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the old value. If you delete a resource, any tags associated with the resource are also deleted.

## Tag Restrictions and Limitations

The following basic restrictions apply to tags:

- Maximum number of tags per resource — 50
- Maximum key length — 127 Unicode characters in UTF-8
- Maximum value length — 255 Unicode characters in UTF-8
- Tag keys and values are case sensitive.
- Do not use the "aws:" prefix in your tag names or values. It's reserved for AWS use. You can't edit or delete tag names or values with this prefix. Tags with this prefix don't count against your tags per resource limit.
- If your tagging schema is used across multiple services and resources, remember that other services might have restrictions on allowed characters. Allowed characters include letters, spaces, and numbers representable in UTF-8, and the following special characters: + - = . \_ : / @.

## Using Tags with IAM Policies

You can apply tag-based resource-level permissions in the IAM policies you use for AWS IoT API actions. This gives you better control over what resources a user can create, modify, or use. You use the Condition element (also called the Condition block) with the following condition context keys and values in an IAM policy to control user access (permissions) based on a resource's tags:

- Use `aws:ResourceTag/tag-key: tag-value` to allow or deny user actions on resources with specific tags.
- Use `aws:RequestTag/tag-key: tag-value` to require that a specific tag be used (or not used) when making an API request to create or modify a resource that allows tags.
- Use `aws:TagKeys: [tag-key, ...]` to require that a specific set of tag keys be used (or not used) when making an API request to create or modify a resource that allows tags.

### Note

The condition context keys and values in an IAM policy apply only to those AWS IoT actions where an identifier for a resource capable of being tagged is a required parameter. For example, the use of [DescribeEndpoint](#) is not allowed or denied on the basis of condition context keys and values because no taggable resource (thing groups, thing types, topic rules, jobs, or security profile) is referenced in this request.

For more information about using tags, see [Controlling Access Using Tags](#) in the *AWS Identity and Access Management User Guide*. The [IAM JSON Policy Reference](#) section of that guide has detailed syntax, descriptions, and examples of the elements, variables, and evaluation logic of JSON policies in IAM.

The following example policy applies two tag-based restrictions. An IAM user restricted by this policy:

- Cannot give a resource the tag "env=prod" (in the example, see the line "aws:RequestTag/env" : "prod")

- Cannot modify or access a resource that has an existing tag "env=prod" (in the example, see the line "aws:ResourceTag/env" : "prod").

```
{
    "Version" : "2012-10-17",
    "Statement" : [
        {
            "Effect" : "Deny",
            "Action" : "iot:/*",
            "Resource" : "*",
            "Condition" : {
                "StringEquals" : {
                    "aws:RequestTag/env" : "prod"
                }
            }
        },
        {
            "Effect" : "Deny",
            "Action" : "iot:/*",
            "Resource" : "*",
            "Condition" : {
                "StringEquals" : {
                    "aws:ResourceTag/env" : "prod"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:/*"
            ],
            "Resource": "*"
        }
    ]
}
```

You can also specify multiple tag values for a given tag key by enclosing them in a list, like this:

```
"StringEquals" : {
    "aws:ResourceTag/env" : ["dev", "test"]
}
```

#### Note

If you allow or deny users access to resources based on tags, you must consider explicitly denying users the ability to add those tags to or remove them from the same resources. Otherwise, it's possible for a user to circumvent your restrictions and gain access to a resource by modifying its tags.

## Billing Groups

AWS IoT doesn't allow you to directly apply tags to individual things, but it does allow you to place things in billing groups and to apply tags to these. For AWS IoT, allocation of cost and usage data based on tags is limited to billing groups.

The following commands are available:

- [AddThingToBillingGroup](#) adds a thing to a billing group.

- [CreateBillingGroup](#) creates a billing group.
- [DeleteBillingGroup](#) deletes the billing group.
- [DescribeBillingGroup](#) returns information about a billing group.
- [ListBillingGroups](#) lists the billing groups you have created.
- [ListThingsInBillingGroup](#) lists the things you have added to the given billing group.
- [RemoveThingFromBillingGroup](#) removes the given thing from the billing group.
- [UpdateBillingGroup](#) updates information about the billing group.
- [CreateThing](#) allows you to specify a billing group for the thing when you create it.
- [DescribeThing](#) returns the description of a thing including the billing group the thing belongs to, if any.

## Viewing Cost Allocation and Usage Data

You can use billing group tags to categorize and track your costs. When you apply tags to billing groups (and so to the things they include), AWS generates a cost allocation report as a comma-separated value (CSV) file with your usage and costs aggregated by your tags. You can apply tags that represent business categories (such as cost centers, application names, or owners) to organize your costs across multiple services. For more information about using tags for cost allocation, see [Use Cost Allocation Tags](#) in the [AWS Billing and Cost Management User Guide](#).

### Note

To accurately associate usage and cost data with those things you have placed in billing groups, each device or application must:

- Be registered as a thing in AWS IoT. For more information, see [Managing Devices with AWS IoT \(p. 93\)](#).
- Connect to the AWS IoT message broker through MQTT using only the thing's name as the client ID. For more information, see [Message Broker for AWS IoT \(p. 244\)](#).
- Authenticate using a client certificate associated with the thing.

The following pricing dimensions are available for billing groups (based on the activity of things associated with the billing group):

- Connectivity (based on the thing name used as the client ID to connect).
- Messaging (based on messages inbound from, and outbound to, a thing; MQTT only).
- Shadow operations (based on the thing whose message triggered a shadow update).
- Rules triggered (based on the thing whose inbound message triggered the rule; does not apply to those rules triggered by MQTT lifecycle events).
- Thing index updates (based on the thing that was added to the index).
- Remote actions (based on the thing updated).
- [Detect \(p. 635\)](#) reports (based on the thing whose activity is reported).

Cost and usage data based on tags (and reported for a billing group) doesn't reflect the following activities:

- Device registry operations (including updates to things, thing groups, and thing types). For more information, see [Managing Devices with AWS IoT \(p. 93\)](#).
- Thing group index updates (when adding a thing group).
- Index search queries.
- [Device Provisioning \(p. 496\)](#).

- [Audit \(p. 534\)](#) reports.

# Security in AWS IoT

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS compliance programs](#). To learn about the compliance programs that apply to AWS IoT, see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

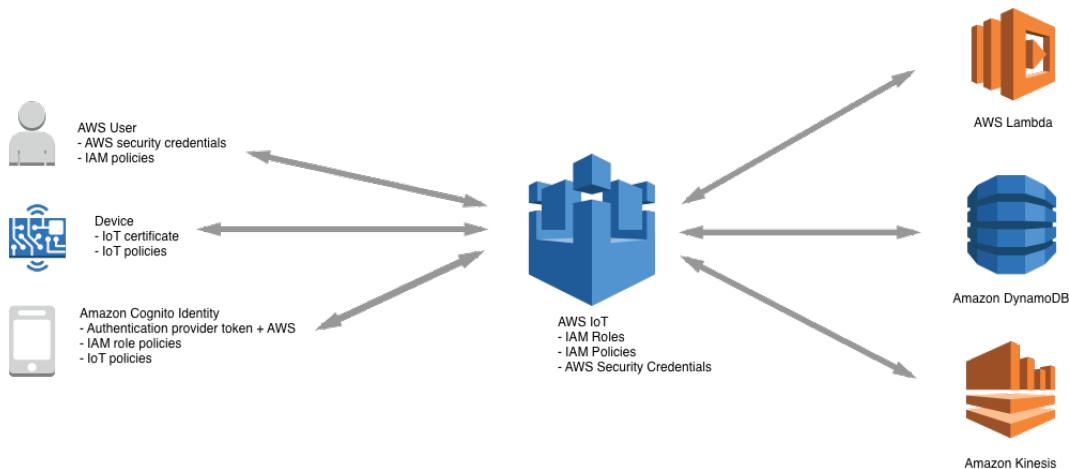
This documentation helps you understand how to apply the shared responsibility model when using AWS IoT. The following topics show you how to configure AWS IoT to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS IoT resources.

## Topics

- [AWS IoT Security \(p. 119\)](#)
- [Authentication \(p. 120\)](#)
- [Managing Device Certs \(p. 137\)](#)
- [Authorization \(p. 137\)](#)
- [Data Protection in AWS IoT Core \(p. 172\)](#)
- [Identity and Access Management for AWS IoT \(p. 174\)](#)
- [Logging and Monitoring with AWS IoT Core \(p. 197\)](#)
- [Compliance Validation for AWS IoT Core \(p. 233\)](#)
- [Resilience in AWS IoT Core \(p. 234\)](#)
- [Infrastructure Security in AWS IoT \(p. 234\)](#)
- [Vulnerability Analysis and Management in AWS IoT Core \(p. 234\)](#)
- [Security Best Practices in AWS IoT Core \(p. 235\)](#)
- [AWS Training and Certification \(p. 238\)](#)

## AWS IoT Security

Each connected device or client must have a credential to interact with AWS IoT. All traffic to and from AWS IoT is sent securely over Transport Layer Security (TLS). AWS cloud security mechanisms protect data as it moves between AWS IoT and other AWS services.



- You are responsible for managing device credentials (X.509 certificates, AWS credentials, Amazon Cognito identities, federated identities, or custom authentication tokens) and policies in AWS IoT. For more information, see [Key Management in AWS IoT \(p. 174\)](#). You are responsible for assigning unique identities to each device and managing the permissions for each device or group of devices.
- Your devices connect to AWS IoT using X.509 certificates or Amazon Cognito identities over a secure TLS connection. During research and development, and for some applications that make API calls or use WebSockets, you can also authenticate using IAM users and groups or custom authentication tokens. For more information, see [IAM Users, Groups, and Roles \(p. 128\)](#).
- When using AWS IoT authentication, the message broker is responsible for authenticating your devices, securely ingesting device data, and granting or denying access permissions you specify for your devices using AWS IoT policies.
- When using custom authentication, a custom authorizer is responsible for authenticating your devices and granting or denying access permissions you specify for your devices using AWS IoT or IAM policies.
- The AWS IoT rules engine forwards device data to other devices or other AWS services according to rules you define. It uses AWS Identity and Access Management to securely transfer data to its final destination. For more information, see [Identity and Access Management for AWS IoT \(p. 174\)](#).

## Authentication

Authentication is a mechanism where you verify the identity of a client or a server. Server authentication is the process where devices or other clients ensure they are communicating with an actual AWS IoT endpoint. Client authentication is the process where devices or other clients authenticate themselves with AWS IoT.

## AWS Training and Certification

Take the following course to learn about authentication in AWS IoT: [Deep Dive into AWS IoT Authentication and Authorization](#).

## X.509 Certificate Overview

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. X.509 certificate chains are used both for server authentication by clients and client authentication by the server.

## Server Authentication

When your device or other client attempts to connect to AWS IoT Core, the AWS IoT Core server will send an X.509 certificate that your device uses to authenticate the server. Authentication takes place at the TLS layer through validation of the [X.509 certificate chain](#). This is the same method used by your browser when you visit an HTTPS URL.

When your devices or other clients establish a TLS connection to an AWS IoT Core endpoint, AWS IoT Core presents a certificate chain that the devices use to verify that they're communicating with AWS IoT Core and not another server impersonating AWS IoT Core. The chain that is presented depends on a combination of the type of endpoint the device is connecting to and the [cipher suite](#) that the client and AWS IoT Core negotiated during the TLS handshake.

## Endpoint Types

AWS IoT Core supports two different data endpoint types, `iot:Data` and `iot:Data-ATS`. `iot:Data` endpoints present a certificate signed by the [VeriSign Class 3 Public Primary G5 root CA certificate](#). `iot:Data-ATS` endpoints present a server certificate signed by an [Amazon Trust Services](#) CA.

Certificates presented by ATS endpoints are cross signed by Starfield. Some TLS client implementations require validation of the root of trust and require that the Starfield CA certificates are installed in the client's trust stores.

### Warning

Using a method of certificate pinning that hashes the whole certificate (including the issuer name, and so on) is not recommended because this will cause certificate verification to fail because the ATS certificates we provide are cross signed by Starfield and have a different issuer name.

Use `iot:Data-ATS` endpoints unless your device requires Symantec or Verisign CA certificates. Symantec and Verisign certificates have been deprecated and are no longer supported by most web browsers.

You can use the `describe-endpoint` command to create your ATS endpoint.

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

The `describe-endpoint` command returns an endpoint in the following format.

```
account-specific-prefix.iot.your-region.amazonaws.com
```

The first time `describe-endpoint` is called, an endpoint is created. All subsequent calls to `describe-endpoint` return the same endpoint.

For backward-compatibility, AWS IoT Core still supports Symantec endpoints. For more information, see [How AWS IoT Core is Helping Customers Navigate the Upcoming Distrust of Symantec Certificate Authorities](#). Devices operating on ATS endpoints are fully interoperable with devices operating on Symantec endpoints in the same account and do not require any re-registration.

### Note

To see your `iot:Data-ATS` endpoint in the AWS IoT Core console, choose **Settings**. The console displays only the `iot:Data-ATS` endpoint. By default, the `describe-endpoint` command displays the `iot:Data` endpoint for backward compatibility. To see the `iot:Data-ATS` endpoint, specify the `--endpointType` parameter, as in the previous example.

## Creating an `IotDataPlaneClient` with the AWS SDK for Java

By default, the [AWS SDK for Java - Version 2](#) creates an `IotDataPlaneClient` by using an `iot:Data` endpoint. To create a client that uses an `iot:Data-ATS` endpoint, you must do the following.

- Create an `iot:Data-ATS` endpoint by using the [DescribeEndpoint API](#).
- Specify that endpoint when you create the `IotDataPlaneClient`.

The following example performs both of these operations.

```
public void setup() throws Exception {
    IotClient client =
    IotClient.builder().credentialsProvider(CREDENTIALS_PROVIDER_CHAIN).region(Region.US_EAST_1).build();
    String endpoint = client.describeEndpoint(r -> r.endpointType("iot:Data-
ATS")).endpointAddress();
    iot = IotDataPlaneClient.builder()
        .credentialsProvider(CREDENTIALS_PROVIDER_CHAIN)
        .endpointOverride(URI.create("https://" + endpoint))
        .region(Region.US_EAST_1)
        .build();
}
```

## CA Certificates for Server Authentication

Depending on which type of data endpoint you are using and which cipher suite you have negotiated, AWS IoT Core server authentication certificates are signed by one of the following root CA certificates:

### VeriSign Endpoints (legacy)

- RSA 2048 bit key: [VeriSign Class 3 Public Primary G5 root CA certificate](#)

### Amazon Trust Services Endpoints (preferred)

- RSA 2048 bit key: [Amazon Root CA 1](#).
- RSA 4096 bit key: Amazon Root CA 2. Reserved for future use.
- ECC 256 bit key: [Amazon Root CA 3](#).
- ECC 384 bit key: Amazon Root CA 4. Reserved for future use.

These certificates are all cross-signed by the [Starfield Root CA Certificate](#). All new AWS IoT Core regions, beginning with the May 9, 2018 launch of AWS IoT Core in the Asia Pacific (Mumbai) Region, serve only ATS certificates.

## Server Authentication Guidelines

There are many variables that can affect a device's ability to validate the AWS IoT Core server authentication certificate. For example, devices may be too memory constrained to hold all possible root CA certificates, or devices may implement a non-standard method of certificate validation. For these reasons we suggest following these guidelines:

- We recommend that you use your ATS endpoint and install all supported Amazon Root CA certificates.
- If you cannot store all of these certificates on your device and if your devices do not use ECC-based validation, you can omit the [Amazon Root CA 3](#) and [Amazon Root CA 4](#) ECC certificates. If your devices do not implement RSA-based certificate validation, you can omit the [Amazon Root CA 1](#) and [Amazon Root CA 2](#) RSA certificates.
- If you are experiencing server certificate validation issues when connecting to your ATS endpoint, try adding the relevant cross-signed Amazon Root CA certificate to your trust store.
  - [Cross-signed Amazon Root CA 1](#)
  - [Cross-signed Amazon Root CA 2](#) - Reserved for future use.

- [Cross-signed Amazon Root CA 3](#)
- [Cross-signed Amazon Root CA 4 - Reserved for future use.](#)
- If you are experiencing server certificate validation issues, your device may need to explicitly trust the root CA. Try adding the [Starfield Root CA Certificate](#) to your trust store.
- If you still experience issues after executing the steps above, please contact [AWS Developer Support](#).

**Note**

CA certificates have an expiration date after which they cannot be used to validate a server's certificate. CA certificates might have to be replaced before their expiration date. Make sure 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.

**Note**

When connecting to AWS IoT Core in your device code, pass the certificate into the API you are using to connect. The API you use will vary by SDK. For more information, see the [AWS IoT Core Device SDKs](#).

## Client Authentication

AWS IoT supports three types of identity principals for device or client authentication:

- [X.509 Client Certificates \(p. 123\)](#)
- [IAM Users, Groups, and Roles \(p. 128\)](#)
- [Amazon Cognito Identities \(p. 129\)](#)

These identities can be used with devices, mobile, web, or desktop applications. They can even be used by a user typing AWS IoT command line interface (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. For more information about IAM identities, see [Identity and Access Management for AWS IoT \(p. 174\)](#).

### X.509 Client Certificates

X.509 certificates provide several benefits over other identification and authentication mechanisms. X.509 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. X.509 certificates provide stronger client authentication over other schemes, such as user name and password or bearer tokens, because the private key never leaves the device.

AWS IoT authenticates client 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 an X.509 client 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.

AWS IoT supports the following client certificate-signing algorithms:

- SHA256WITHRSA
- SHA384WITHRSA
- SHA512WITHRSA
- DSA\_WITH\_SHA256
- ECDSA-WITH-SHA256
- ECDSA-WITH-SHA384

- ECDSA-WITH-SHA512

AWS IoT can use AWS IoT-generated X.509 certificates or your own X.509 certificates for device authentication. X.509 certificates generated by AWS IoT are long-lived (but will expire at 2049-12-31T23:59:59Z, that is at midnight GMT on December 31, 2049.) The expiry date and time for certificates signed by a CA certificate are set when the certificate is created.

To use an X.509 certificate not created by AWS IoT, you must register a CA certificate. All device certificates must be signed by the CA certificate you register.

**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.

You can use the AWS IoT console or CLI to perform the following certificate 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 Register an AWS IoT Device Certificate \(p. 124\)](#).

For more information about using your own X.509 certificates, see [Use Your Own Certificate \(p. 125\)](#).

## Create and Register an AWS IoT Device Certificate

You can use the AWS IoT console or the AWS 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](#).
2. In the left navigation pane, choose **Security**, choose **Certificates**, and then choose **Create**.
3. Choose **One-click certificate creation - Create certificate**. Or to generate a certificate with a certificate signing request (CSR), choose **Create with CSR**.
4. Download the public key, private key, and certificate to a secure location.
5. Choose **Activate**.
6. Choose **Done**. You can also choose **Attach a policy** to attach an existing AWS IoT policy to the certificate.

### To create a certificate (CLI)

The AWS CLI provides two commands to create certificates:

- [create-keys-and-certificate](#)

This command creates a private key, public key, and X.509 certificate.

- [create-certificate-from-csr](#)

This command creates a certificate given a CSR.

## Use Your Own Certificate

To use your own X.509 device 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 AWS 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.

### Topics

- [Registering Your CA Certificate \(p. 125\)](#)
- [Creating a Device Certificate Using Your CA Certificate \(p. 126\)](#)
- [Registering a Device Certificate \(p. 127\)](#)
- [Registering Device Certificates Manually \(p. 127\)](#)
- [Using Automatic/Just-in-Time Registration for Device Certificates \(p. 127\)](#)
- [Deactivate a CA Certificate \(p. 128\)](#)
- [Revoke a Device Certificate \(p. 128\)](#)

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

### Note

A CA certificate cannot be registered to more than one account in the same AWS Region. However, a CA certificate can be registered to more than one account if the accounts are in different AWS Regions.

### To register a CA certificate

1. Get a registration code from AWS IoT. This code is 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 are 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 -Ccreateserial -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 xxxxxxxxxxxx --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 are 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 -Ccreateserial -out deviceCert.pem -days 500 -sha256**

**Note**

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 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 xxxxxxxxxxxx --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 **register-certificate** 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 registers any device certificate signed by your CA certificate when it connects to AWS IoT.

#### Enable Automatic Registration

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

```
aws iot update-ca-certificate --certificate-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 attempts to connect to AWS IoT for the first time, as part of the TLS handshake, it must present a file that contains both your registered CA certificate and the device certificate signed by your CA certificate. You can combine the two files using a command like the following:

```
cat deviceCert.crt sampleCACertificate.pem > deviceCertAndCACert.crt
```

When you connect to AWS IoT, use the `deviceCertAndCACert.crt` file as your certificate file. AWS IoT recognizes the CA certificate as a registered CA certificate, registers the device certificate, and sets its status to `PENDING_ACTIVATION`. This means that 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 device presents a certificate in the `PENDING_ACTIVATION` status, 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 can 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](#).

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](#).

### Deactivate a CA Certificate

When you register a device certificate, AWS IoT checks if the CA certificate used to sign the device 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 CA from being registered in your account. You can use the **update-ca-certificate** CLI command to deactivate the CA certificate:

```
aws iot update-ca-certificate --certificate-id certificateID --new-status INACTIVE
```

#### Note

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

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

### Revoke a Device Certificate

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

```
aws iot update-certificate --certificate-id certificateID --new-status REVOKED
```

## 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 [Identity and Access Management for AWS IoT \(p. 174\)](#).

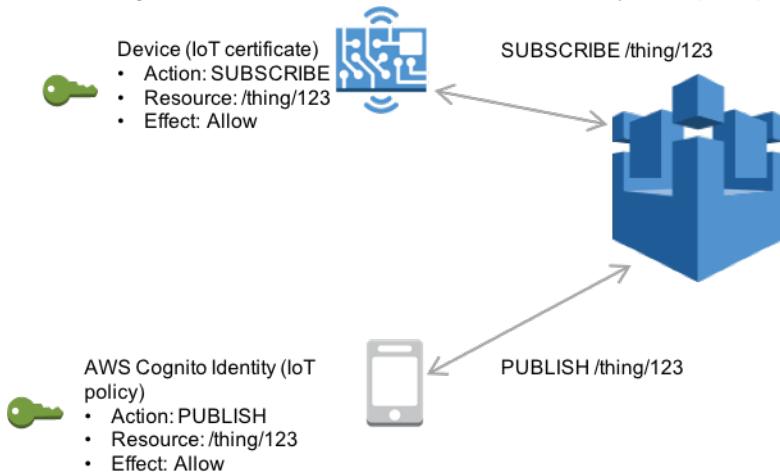
## Amazon Cognito Identities

Amazon Cognito Identity enables you to create temporary, limited-privilege AWS credentials for use in mobile and web applications. When you use Amazon Cognito Identity, you create identity pools that create unique identities for your users and authenticate them with identity providers like Login with Amazon, Facebook, and Google. You can also use Amazon Cognito identities with your own developer authenticated identities. For more information, see [Amazon Cognito Identity](#).

To use Amazon Cognito Identity, you define a Amazon Cognito identity pool that is associated with an IAM role. The IAM role is associated with an IAM policy that grants identities from your identity pool permission to access AWS resources like calling AWS services.

Amazon Cognito Identity creates unauthenticated and authenticated identities. Unauthenticated identities are used for guest users in a mobile or web application who want to use the app without signing in. Unauthenticated users are granted only those permissions specified in the IAM policy associated with the identity pool.

When you use authenticated identities, in addition to the IAM policy attached to the identity pool, you can attach an AWS IoT policy to an Amazon Cognito Identity using the [AttachPolicy](#) API and give fine-grained permissions to an individual user of your AWS IoT application. In this way, you can assign permissions for specific customers and their devices. For more information about creating policies for Amazon Cognito identities, see [Publish/Subscribe Policy Examples \(p. 150\)](#).



## Custom Authentication

AWS IoT lets you define custom authorizers so that you can manage your own device authentication and authorization. To do so, you use AWS Lambda functions to send device credentials to your authentication service.

When an HTTP connection is established (and, optionally, upgraded to a WebSocket connection) and *Signature Version 4* headers aren't present, the AWS IoT device gateway checks if a custom authorizer is configured for the endpoint. If so, AWS IoT device gateway uses the custom authorizer to authenticate the connection and authorize the device. Custom authorizers can implement various authentication strategies (for example, JWT verification, OAuth provider callout, and so on) and must return policy documents that are used by the device gateway to authorize MQTT operations.

There are two ways to implement custom authentication:

- **Custom authentication** – Uses HTTP Publish operations or MQTT over WSS connections to receive device credentials in a bearer token (such as a JSON Web Token (JWT) or OAuth token) inside an HTTP header.
- **Enhanced custom authentication** – Extends custom authentication to enable passing device credentials through an MQTT CONNECT message (in the `username` and `password` fields) or as query parameters in an HTTP Publish or Upgrade request (in the case of MQTT over WSS). This feature also enables you to remove the token signing requirements for your custom authorizers. This features is beta.

**Note**

You can define up to 10 custom authorizers.

**Topics**

- [Custom Authorizers \(p. 130\)](#)
- [Configure a Custom Authorizer \(p. 132\)](#)
- [Custom Authorizer Workflow \(p. 133\)](#)
- [Enhanced Custom Authentication \(Beta\) \(p. 134\)](#)

## Custom Authorizers

Custom authorizers consist of:

Name

A unique arbitrary string that identifies the authorizer.

Lambda function ARN

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

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**

You are charged based on the number of requests for your Lambda functions and the duration, the time it takes for your code to execute. For more information about AWS Lambda, see [AWS Lambda Pricing](#) and [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.

**principalId**

The ID of the principal that is granted the permissions described in `policyDocuments`.

**policyDocuments**

A list of JSON formatted policy documents following the same conventions as an AWS IoT policy.

The list contains at most 10 policy documents, each of which can be a maximum of 2,048 characters.

**DisconnectAfterInSecs**

The maximum duration (in seconds) of the connection to the AWS IoT gateway, after which it is disconnected. The minimum value is 300 seconds, and the maximum value is 86,400 seconds.

**RefreshAfterInSecs**

The period between policy refreshes. When it lapses, the AWS policy engine reevaluates the policy documents and the AWS IoT Device Gateway invokes the Lambda function again to allow for policy changes. The minimum value is 300 seconds, and the maximum value is 86,400 seconds.

**Context**

Information derived after validating the token that 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 <lambda_function_name> --statement-id  
<unique_identifier_string> --action 'lambda:InvokeFunction' --principal iot.amazonaws.com --  
source-arn arn:aws:iot:<your-aws-region>:<account_id>: authorizer/<authorizer-name>
```

The **add-permission** CLI command takes the following parameters:

**function-name**

The name of the Lambda function to which you are granting invocation permission.

**statement-id**

A statement identifier.

**action**

The Lambda action you are granting permission to perform.

**principal**

The principal to which you are granting permission.

#### source-arn

The ARN of the custom authorizer. Specifying this value ensures your Lambda function can be invoked only by the intended custom authorizer.

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>
```

## Configure a Custom Authorizer

1. Create a Lambda function that implements your authentication/authorization logic (for example, the `MyAuthorizerFunction` in the following step). The following is an example of what a custom authorizing Lambda function might return:

```
{
    "isAuthenticated": true,
    "principalId": "xxxxxxxxxx",
    "disconnectAfterInSeconds": 86400,
    "refreshAfterInSeconds": 300,
    "policyDocuments": [
        "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Action\":\"...\", \"Effect\":
\"Allow/Deny\", \"Resource\":\"...\"}]}"
    ],
    "context": {
        "username": "johnDoe123",
        "city": "Seattle",
        "country": "USA"
    }
}
```

The return value of the Lambda function should be similar to this response. It can be either a JSON serialized or non-serialized object.

2. Use the `create-authorizer` API to register a custom authorizer with AWS IoT.

```
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
```

You can use the `test-invoke-authorizer` API to test if the custom authorizer Lambda function has been configured correctly, as shown:

```
aws iot test-invoke-authorizer --authorizer-name <NAME_OF_AUTHORIZER> --token
<TOKEN_VALUE> --token-signature <TOKEN_SIGNATURE>
```

**Note**

`<TOKEN_SIGNATURE>` must be signed with the private key of the public-private key pair uploaded to AWS IoT used in the `create-authorizer` call. One method of locally creating `<TOKEN_SIGNATURE>` from a UNIX-like command line is as follows:

```
echo -n <TOKEN_VALUE> | openssl dgst -sha256 -sign <PRIVATE_KEY> | openssl base64
```

You must trim all newline characters from the result of this command before passing the `<TOKEN_SIGNATURE>` value to the `test-invoke-authorizer` API.

## 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>
sec-WebSocket-Key: <any random base64 value>
sec-websocket-protocol: mqtt
sec-WebSocket-Version: <websocket version>
```

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.

**Note**

Some web browsers might not support custom HTTP headers.

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.

```
{
  "token": "some-token"
}
```

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

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

```
{  
    "isAuthenticated":true,  
    "principalId": "xxxxxxxx",  
    "disconnectAfterInSeconds": 86400,  
    "refreshAfterInSeconds", 300,  
    "policyDocuments": [  
        {"Version": "2012-10-17", "Statement": [ { "Action": "\u2026", "Effect":  
        "Allow/Deny", "Resource": "\u2026" } ] }  
    ]  
}
```

The return value of the Lambda function should be similar to this response and can be either a JSON serialized or non-serialized object.

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.

For an end-to-end example of this workflow, see [How to Use Your Own Identity and Access Management Systems to Control Access to AWS IoT Resources](#).

## Enhanced Custom Authentication (Beta)

This feature is currently in public beta and is available only in the US East (N. Virginia) Region.

Enhanced custom authentication provides greater flexibility in the authentication methods that your custom authorizers can use when the custom authorizers are implemented by configurable endpoints. This set of preview features enables you to pass tokens to your authorizers by using HTTP query strings in addition to headers. It also makes token signing optional. These features also enable authentication by passing credentials in the `username` and `password` fields of an MQTT CONNECT message. This topic describes how you can use these enhancements and how to add custom authorizers to your configurable endpoints. For more information, see [Configurable Endpoints \(Beta\)](#).

Enhanced custom authentication sends device credentials to your custom authorizers by using one of the following methods:

- HTTP Publish (header fields)
- HTTP Publish (query string)
- MQTT Connect (user name and password fields)
- HTTP Upgrade to MQTT (HTTP headers or query string)
- HTTP Upgrade to MQTT (MQTT user name and password)

### Note

To pass unsigned tokens or use MQTT user name/password authentication, devices must send the Server Name Indication (SNI) TLS extension with a fully qualified domain name that corresponds to the name specified in the domain configuration. To test this service, use the v2 version of each [AWS IoT device SDK in GitHub](#).

### Topics

- [Creating a Custom Authorizer for Enhanced Custom Authentication \(p. 135\)](#)
- [Invoking a Custom Authorizer with Enhanced Custom Authentication \(p. 135\)](#)

## Creating a Custom Authorizer for Enhanced Custom Authentication

This feature is currently in public beta and is available only in the US East (N. Virginia) Region.

You create a custom authorizer for enhanced custom authentication using the same [CreateAuthorizer](#) API that you use to create a custom authorizer for custom authentication. For procedures to create a custom authorizer, see [Configure a Custom Authorizer](#).

You can use the `signingDisabled` parameter of the `CreateAuthorizer` API. This lets you opt out of signature validation by AWS IoT against the token provided in your HTTP requests. We strongly recommend that you use this feature only in test environments. You can't update the `signingDisabled` value of a custom authorizer. To change this behavior, you must create a custom authorizer with a different value for `signingDisabled`. Values for `tokenKeyName` and `tokenSigningPublicKeys` are optional, but `tokenKeyName` is required when you specify a value for `tokenSigningPublicKeys`.

### Adding a Custom Authorizer to a Configurable Endpoint

After you create your custom authorizer, you can attach it to an existing domain configuration as the default authorizer by using the [UpdateDomainConfiguration](#) API. You can also create a configuration by using the [CreateDomainConfiguration](#) API and specifying the name of your custom authorizer in the `defaultAuthorizerName` parameter.

## Invoking a Custom Authorizer with Enhanced Custom Authentication

This feature is currently in public beta and is available only in the US East (N. Virginia) Region.

This section explains how to use enhanced custom authentication to pass device credentials to your custom authorizer by using HTTP headers and query strings or MQTT user names and passwords.

Devices that use enhanced custom authentication must send the Server Name Indication (SNI) TLS extension with a value that matches the domain of the appropriate domain configuration. You can specify a custom authorizer that isn't the default authorizer if the `allowAuthorizerOverride` value in your domain configuration is set to `true`.

To use enhanced custom authentication in MQTT connections, devices must also send the Application Layer Protocol Negotiation (ALPN) TLS extension with a value of `mqtt` and connect on port 443.

#### Note

You can find configuration for these TLS extensions in the V2 AWS IoT Device SDKs. For more information, see the [AWS Labs GitHub repo](#).

### HTTPS Requests

Requests to custom authorizers can pass tokens by using one of the following: HTTP headers or query strings in HTTP Publish requests or HTTP Upgrade requests for establishing MQTT over WSS sessions. The following example uses HTTP headers to send an upgrade request to AWS IoT Device Gateway.

```
GET /mqtt HTTP/1.1
Host: your-endpoint
Upgrade: WebSocket
Connection: Upgrades
x-amz-customauthorizer-signature: token-signature
token-key-name: some-token
sec-WebSocket-Key: any random base64 value
sec-websocket-protocol: mqtt
```

```
sec-WebSocket-Version: websocket version
```

The request doesn't contain a value for `x-amz-customauthorizer-name`, so AWS IoT uses the default authorizer specified in the domain configuration. The value of `x-amz-customauthorizer-signature` is optional if signing is disabled on the authorizer.

The following example shows how to make the same request by using query string parameters.

```
GET /mqtt?x-amz-customauthorizer-signature=${sign}&token-name=${token-value} HTTP/1.1
Host: your-endpoint
Connection: Upgrade
Upgrade: websocket
sec-WebSocket-Key: any random base64 value
sec-websocket-protocol: mqtt
sec-WebSocket-Version: websocket version
```

## MQTT Username and Password

You can pass user names and passwords to your custom authorizers by using the `username` and `password` fields of MQTT messages. Password data is base64-encoded to support arbitrary binary values. The `username` value can optionally contain a query string that passes additional values (including a token, signature, and authorizer name) to your authorizer.

The following example contains a `username` string with extra parameters that specify a token and signature. You can use this method to authenticate an MQTT connection by using a bearer token.

```
username?x-amz-customauthorizer-name=${name}&x-amz-customauthorizer-signature=
${sign}&token-name=${token-value}
```

## Data Sent to the Custom Authorizer

The data that AWS IoT sends to your custom authorizer Lambda function depends on which protocols and parameters are present in the connection. For HTTP requests (Publishes and WSS Upgrades), AWS IoT sends all headers and query parameters (up to 8 KB of data). For WSS upgrades on MQTT Connect, AWS IoT also sends all MQTT data.

If signing is enabled and the request contains a token and signature, AWS IoT won't trigger your custom authorizer's Lambda function unless the token signature can be decrypted to match the token provided in the request. You can use this method so that others connecting to your domain don't excessively trigger your Lambda function.

### Important

We strongly recommend that you use the signature verification feature that AWS IoT offers.

The following is an example payload that AWS IoT sends to the custom authenticator's Lambda function. AWS IoT populates only the fields that the request contains.

```
{
  "token" : "aToken",
  "signatureVerified": boolean, // Indicates whether the device gateway has validated the
                                // signature.
  "protocols": ["tls", "http", "mqtt"], // Indicates which protocols to expect.
  "protocolData": {
    "tls" : {
      "serverName": "serverName" // The SNI string.
    },
  }
}
```

```
    "http": {
        "headers": {
            "#${name}": "#{value}"
        },
        "queryString": "?#{name}=#${value}"
    },
    "mqtt": {
        "username": "myUserName",
        "password": "myPassword", // base64 encoded.
        "clientId": "myClientId" // Provided only when the device sends it.
    }
},
"connectionMetadata": {
    "id": Uuid // The connection ID. You can use this for logging.
},
}
```

#### Note

All password data is base64 encoded and your Lambda function needs to decode it.

As with the regular custom authentication, the custom authorizer must return a response that indicates whether it has authorized the request to AWS IoT. The following is an example of a successful response from a custom authorizer.

```
{
    "isAuthenticated":true,
    "principalId": "xxxxxxxxxx",
    "disconnectAfterInSeconds": 86400,
    "refreshAfterInSeconds", 300,
    "policyDocuments": [
        "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Action\": \"...\", \"Effect\": \"Allow|Deny\", \"Resource\": \"...\" } ] }"
    ]
}
```

AWS IoT caches the policies associated with the principal for the duration specified in the value of `refreshAfterInSeconds`. During this duration, subsequent calls are authorized without having to reauthenticate the device. Failures that occurs during custom authentication results in authentication failure, which stops the connection.

## Managing Device Certs

Your devices can use X.509 certificates to authenticate with AWS IoT Core.

### Server Authentication

The AWS IoT root CA certificate allows your devices to verify that they're communicating with AWS IoT Core and not another server impersonating AWS IoT Core. For more information, see [CA Certificates for Service Authentication](#).

## Authorization

Authorization is the process of granting permissions to an authenticated identity. You grant permissions in AWS IoT Core using AWS IoT Core and IAM policies. This topic covers AWS IoT Core policies. For more

information about IAM policies, see [Identity and Access Management for AWS IoT \(p. 174\)](#) and [IAM Policies \(p. 179\)](#).

AWS IoT Core 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 Core CLI commands. An identity can execute AWS IoT Core operations only if it has a policy that grants it permission for those operations.

Both AWS IoT Core policies and IAM policies are used with AWS IoT Core 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 Core.

AWS IoT Core 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 Core.

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

The following table shows the identity types, the protocols they use, and the policy types that can be used for authorization.

### AWS IoT Core Data Plane API and Policy Types

| Protocol and Authentication Mechanism                                                 | SDK                     | Identity Type                              | Policy Type                   |  |  |
|---------------------------------------------------------------------------------------|-------------------------|--------------------------------------------|-------------------------------|--|--|
| MQTT over TLS/TCP, TLS mutual authentication (port 8883 or 443) <sup>† (p. 253)</sup> | AWS IoT Core Device SDK | X.509 certificates                         | AWS IoT Core policy           |  |  |
| MQTT over HTTPS/ WebSocket, AWS SigV4 authentication (port 443)                       | AWS Mobile SDK          | Authenticated Amazon Cognito identity      | IAM and AWS IoT Core policies |  |  |
|                                                                                       |                         | Unauthenticated Amazon Cognito identity    | IAM policy                    |  |  |
|                                                                                       |                         | IAM, or federated identity                 | IAM policy                    |  |  |
| HTTPS, AWS Signature Version 4 authentication (port 443)                              | AWS CLI                 | Amazon Cognito, IAM, or federated identity | IAM policy                    |  |  |
| HTTPS, TLS mutual                                                                     | No SDK support          | X.509 certificates                         | AWS IoT Core policy           |  |  |

| <b>Protocol and Authentication Mechanism</b> | <b>SDK</b>              | <b>Identity Type</b> | <b>Policy Type</b>       |  |  |
|----------------------------------------------|-------------------------|----------------------|--------------------------|--|--|
| authentication (port 8443)                   |                         |                      |                          |  |  |
| HTTPS over custom authentication (Port 443)  | AWS IoT Core Device SDK | Custom authorizer    | Custom authorizer policy |  |  |

### AWS IoT Core Control Plane API and Policy Types

| <b>Protocol and Authentication Mechanism</b>            | <b>SDK</b> | <b>Identity Type</b>       | <b>Policy Type</b> |  |  |
|---------------------------------------------------------|------------|----------------------------|--------------------|--|--|
| HTTPS AWS Signature Version 4 authentication (port 443) | AWS CLI    | Amazon Cognito identity    | IAM policy         |  |  |
|                                                         |            | IAM, or federated identity | IAM policy         |  |  |

AWS IoT Core 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 Core CLI to attach the policy (to a certificate or Amazon Cognito Identity), you use an AWS IoT Core 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 Core. For example, consider a device connecting to AWS IoT Core 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 Core resource for the user. You can also control an application's access to AWS IoT Core resources.

## AWS Training and Certification

For information about authorization in AWS IoT Core, take the [Deep Dive into AWS IoT Core Authentication and Authorization](#) course on the AWS Training and Certification website.

## AWS IoT Core Policies

AWS IoT Core policies are JSON documents. They follow the same conventions as IAM policies. AWS IoT Core 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 Core policies allow you to control access to the AWS IoT Core data plane. The AWS IoT Core data plane consists of operations that allow you to connect to the AWS IoT Core message broker, send and receive MQTT messages, and get or update a device's shadow.

An AWS IoT Core policy is a JSON document that contains one or more policy statements. Each statement contains:

- **Effect**, which specifies whether the action is allowed or denied.
- **Action**, which specifies the action the policy is allowing or denying.
- **Resource**, which specifies the resource or resources on which the action is allowed or denied.

### Topics

- [AWS IoT Core Policy Actions \(p. 140\)](#)
- [AWS IoT Core Action Resources \(p. 141\)](#)
- [AWS IoT Core Policy Variables \(p. 142\)](#)
- [Example AWS IoT Policies \(p. 147\)](#)
- [Authorization with Amazon Cognito Identities \(p. 167\)](#)

## AWS IoT Core Policy Actions

The following policy actions are defined by AWS IoT Core:

### MQTT Policy Actions

`iot:Connect`

Represents the permission to connect to the AWS IoT Core 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 closes the existing connection. The `iot:Connect` permission can be used to ensure only authorized clients using a specific client ID can connect.

`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

To grant `iot:Publish` permission, you must also grant `iot:Connect` permission.

`iot:Receive`

Represents the permission to receive a message from AWS IoT Core. 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

To grant `iot:Subscribe` permission, you must also grant `iot:Connect` permission.

### Shadow Policy Actions

`iot:DeleteThingShadow`

Represents the permission to delete a device's shadow. The `iot:DeleteThingShadow` permission is checked every time a request is made to delete the shadow's contents.

`iot:GetThingShadow`

Represents the permission to retrieve a device's shadow. The `iot:GetThingShadow` permission is checked every time a request is made to retrieve the shadow's contents.

`iot:UpdateThingShadow`

Represents the permission to update a device's shadow. The `iot:UpdateThingShadow` permission is checked every time a request is made to update the shadow's contents.

**Note**

The job execution policy actions apply only for the HTTP TLS endpoint. If you use the MQTT endpoint, you must use MQTT policy actions defined in this topic.

### Job Executions AWS IoT Core Policy Actions

`iot:DescribeJobExecution`

Represents the permission to retrieve a job execution for a given thing. The `iot:DescribeJobExecution` permission is checked every time a request is made to get a job execution.

`iot:GetPendingJobExecutions`

Represents the permission to retrieve the list of jobs that are not in a terminal status for a thing. The `iot:GetPendingJobExecutions` permission is checked every time a request is made to retrieve the list.

`iot:UpdateJobExecution`

Represents the permission to update a job execution. The `iot:UpdateJobExecution` permission is checked every time a request is made to update the state of a job execution.

`iot:StartNextPendingJobExecution`

Represents the permission to get and start the next pending job execution for a thing. (That is, to update a job execution with status QUEUED to IN\_PROGRESS.) The `iot:StartNextPendingJobExecution` permission is checked every time a request is made to start the next pending job execution.

## AWS IoT Core Action Resources

To specify a resource for an AWS IoT Core 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                                                                          |
|------------------------------------|-----------------------------------------------------------------------------------|
| <code>iot:DeleteThingShadow</code> | A thing ARN: <code>arn:aws:iot:us-east-1:123456789012:thing/thingOne</code>       |
| <code>iot:Connect</code>           | A client ID ARN: <code>arn:aws:iot:us-east1:123456789012:client/myClientId</code> |
| <code>iot:Publish</code>           | A topic ARN: <code>arn:aws:iot:us-east-1:123456789012:topic/myTopicName</code>    |

| 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                   |
| iot:DescribeJobExecution         | A thing ARN: arn:aws:iot:us-east-1:123456789012:thing/thingOne                   |
| iot:GetPendingJobExecutions      | A thing ARN: arn:aws:iot:us-east-1:123456789012:thing/thingOne                   |
| iot:UpdateJobExecution           | A thing ARN: arn:aws:iot:us-east-1:123456789012:thing/thingOne                   |
| iot:StartNextPendingJobExecution | A thing ARN: arn:aws:iot:us-east-1:123456789012:thing/thingOne                   |

## AWS IoT Core Policy Variables

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

### Basic AWS IoT Core Policy Variables

AWS IoT Core defines the following basic policy variables:

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

The following AWS IoT Core policy shows a policy that uses 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/filter"
            ]
        }
    ]
}
```

```
        "arn:aws:iot:us-east-1:123451234510:topic/my/topic/${iot:ClientId}"  
    ]  
}  
]
```

In these examples, \${iot:ClientId} is replaced by the ID of the client connected to the AWS IoT Core 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/my/${iot:ClientId}/topic"  
    ]  
}
```

A client can connect using + as the client ID. This would allow the user to subscribe to any topic that matches the topic filter my/+/*topic*. To protect against such security gaps, use the iot:Connect policy action to control which client IDs can connect. For example, this policy allows only those 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"  
            ]  
        }  
    ]  
}
```

## Thing Policy Variables

Thing policy variables allow you to write AWS IoT Core policies that grant or deny permissions based on thing properties like thing names, thing types, and thing attribute values. You can use thing policy variables to apply the same policy to control many AWS IoT Core devices. For more information about device provisioning, see [Device Provisioning](#). The thing name is obtained from the client ID in the MQTT Connect message sent when a thing connects to AWS IoT Core.

Keep the following in mind when using thing policy variables in AWS IoT Core policies.

- Use the [AttachThingPrincipal](#) API to attach certificates or principals (authenticated Amazon Cognito identities) to a thing.
- When you're replacing thing names with thing policy variables, the value of `clientId` in the MQTT connect message or the TLS connection must exactly match the thing name.

The following thing policy variables are available:

- `iot:Connection.Thing.ThingName`

This resolves to the name of the thing in the AWS IoT Core registry for which the policy is being evaluated. AWS IoT Core uses the certificate the device presents when it authenticates to determine

which thing to use to verify the connection. This policy variable is only available when a device connects over MQTT or MQTT over the WebSocket protocol.

- `iot:Connection.Thing.ThingType`

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 certificate or Amazon Cognito identity for which the policy is being evaluated is attached to an IoT thing. You can use this variable to prevent a device from connecting to AWS IoT Core if it presents a certificate that is not attached to an IoT thing in the AWS IoT Core registry.

## X.509 Certificate AWS IoT Core Policy Variables

X.509 certificate policy variables allow you to write AWS IoT Core 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 Core 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 Core 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, but when evaluated in a policy, `iot:Certificate.Subject.Country` is replaced by the first country name. You can request a specific attribute value other than the first by 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 index 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.

## Registered Devices (2)

For devices registered as things in the AWS IoT Core registry, the following policy allows clients with a thing name registered in the AWS IoT Core registry to connect, but restricts the right to publish to a thing name specific topic to those clients with certificates whose `Certificate.Subject.Organization` attribute is set to "Example Corp" or "AnyCompany". This restriction is accomplished by using a "Condition" field that specifies a condition that must be met to allow the preceding action. In this case the condition is that the `Certificate.Subject.Organization` attribute associated with the certificate must include one of the values listed:

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

```

        ],
        "Condition": {
            "ForAllValues:StringEquals": {
                "iot:Certificate.Subject.Organization.List": [
                    "Example Corp",
                    "AnyCompany"
                ]
            }
        }
    ]
}

```

## Unregistered Devices (2)

For devices not registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core with client IDs `client1`, `client2`, and `client3`, but restricts the right to publish to a client-id specific topic to those clients with certificates whose `Certificate.Subject.Organization` attribute is set to "`Example Corp`" or "`AnyCompany`". This restriction is accomplished by using a "`Condition`" field that specifies a condition that must be met to allow the preceding action. In this case the condition is that the `Certificate.Subject.Organization` attribute associated with the certificate must include one of the values listed:

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/my/topic/${iot:ClientId}"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "iot:Certificate.Subject.Organization.List": [
                        "Example Corp",
                        "AnyCompany"
                    ]
                }
            }
        }
    ]
}

```

## Issuer Alternate Name Attributes

The following AWS IoT Core 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 Core 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 Core 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.

## Example AWS IoT Policies

See the following topics to learn about common elements in an AWS IoT policy. You can use the example policies to complete common tasks in AWS IoT.

### Topics

- [AWS IoT Policy Elements \(p. 148\)](#)
- [Connect Policy Examples \(p. 148\)](#)
- [Publish/Subscribe Policy Examples \(p. 150\)](#)

- [Connect and Publish Policy Examples \(p. 162\)](#)
- [Certificate Policy Examples \(p. 162\)](#)
- [Thing Policy Examples \(p. 166\)](#)

## Example Policies

For additional policy examples, see the following topics in other sections of this guide:

- [the section called "Identity-Based Policy Examples" \(p. 193\)](#)
- [the section called "Basic AWS IoT Core Policy Variables" \(p. 142\)](#)
- [the section called "X.509 Certificate AWS IoT Core Policy Variables" \(p. 144\)](#)

## AWS IoT Policy Elements

AWS IoT policies are specified in a JSON document. An AWS IoT policy is composed of the following items:

### *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 device's shadow.  
"iot:GetThingShadow": Retrieve a device's shadow.  
"iot:DeleteThingShadow": Delete a device's 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 grants permission to connect to AWS IoT Core with client ID *client1*:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1"
            ]
        }
    ]
}
```

The following policy denies permission to client IDs `client1` and `client2` to connect to AWS IoT Core, while allowing devices to connect using a client ID that matches the name of a thing registered in the AWS IoT Core registry:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
            ]
        }
    ]
}
```

### Registered Devices (3)

The following policy grants permission for a device to connect using its thing name as the client ID and to subscribe to the topic filter `my/topic/filter`. The device must be registered with AWS IoT Core. When the device connects to AWS IoT Core, it must provide the certificate associated with the IoT thing in the AWS IoT Core registry:

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

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/my/topic/filter"
            ]
        }
    ]
}
```

### Unregistered Devices (3)

For devices not registered as things in the AWS IoT Core registry, the following policy grants permission to connect using client ID `client1` and to subscribe to topic filter `my/topic`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/my/topic"
            ]
        }
    ]
}
```

## Publish/Subscribe Policy Examples

The policy you use depends on how you are connecting to AWS IoT Core. You can connect to AWS IoT Core 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

To specify wildcards in topic names, use `*` in the `resource` attribute of the policy when the device publishes and subscribes to multiple topics. The following policy enables a device to publish to all subtopics that start with the same thing name.

### Registered Devices (5)

For devices registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core using a client ID that matches the thing name and to publish to any topic prefixed by the thing name:

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

#### Unregistered Devices (5)

For devices not registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core using client ID `client1`, `client2`, or `client3` and to publish to any topic prefixed by the client ID:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        }
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/${iot:ClientId}/*"
            ]
        }
    ]
}
```

You can also use the `*` wildcard at the end of a topic filter. Using wildcard characters might lead to granting unintended privileges, so they should only be used after careful consideration. One situation in which they might be useful is when devices must subscribe to messages with many different topics (for example, if a device must subscribe to reports from temperature sensors in multiple locations).

## Registered Devices (6)

For devices registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core using the device's thing name as the client ID, and to subscribe to a topic prefixed by the thing name, followed by `room`, followed by any string. (It is expected that these topics are, for example, `thing1/room1`, `thing1/room2`, and so on):

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/
${iot:Connection.Thing.ThingName}/room*"
            ]
        }
    ]
}
```

## Unregistered Devices (6)

For devices not registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core using client IDs `client1`, `client2`, `client3`, and to subscribe to a topic prefixed by the client ID, followed by `room`, followed by any string. (It is expected that these topics are, for example, `client1/room1`, `client1/room2`, and so on):

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/${iot:ClientId}/room*"
            ]
        }
    ]
}
```

```
    ]  
}
```

When you specify topic filters in AWS IoT Core policies for MQTT clients, MQTT wildcard characters "+" and "#" are treated as literal characters. Their use might result in unexpected behavior.

#### Registered Devices (4)

For devices registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core with the client ID that matches the thing name, and to subscribe to the topic filter `some/+/topic` only:

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

#### Unregistered Devices (4)

For devices not registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core with client ID `client1` and subscribe to the topic filter `some/+/topic` only:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iot:Connect"  
            ],  
            "Resource": [  
                "arn:aws:iot:us-east-1:123456789012:client/client1"  
            ]  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iot:Subscribe"  
            ],  
            "Resource": [  
                "arn:aws:iot:us-east-1:123456789012:topicfilter/some/+/topic"  
            ]  
        }  
    ]  
}
```

```
        "arn:aws:iot:us-east-1:123456789012:topicfilter/some/+/topic"
    ]
}
]
```

### Note

In a policy, the MQTT wildcard character + is treated as a literal, not a wildcard. Attempts to subscribe to topic filters that match the pattern some/+/topic fail and cause the client to disconnect.

## Registered Devices (7)

For devices registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core using the device's thing name as the client ID, and to subscribe to the topics my/topic and my/othertopic:

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

## Unregistered Devices (7)

For devices not registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core using client ID client1, and to subscribe to the topics my/topic and my/othertopic:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/client1"
      ]
    },
  ]
}
```

```
{
    "Effect": "Allow",
    "Action": [
        "iot:Subscribe"
    ],
    "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/my/topic",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/my/othertopic"
    ]
}
```

### Registered Devices (8)

For devices registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core using the device's thing name as the client ID and to subscribe to a topic unique to that thing name/client ID:

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

### Unregistered Devices (8)

For devices not registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core using client ID `client1`, and to publish to a topic unique to that client ID:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1"
            ]
        }
    ]
}
```

```

        ],
    },
{
    "Effect": "Allow",
    "Action": [
        "iot:Publish"
    ],
    "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/my/topic/${iot:ClientId}"
    ]
}
]
}

```

### Registered Devices (9)

For devices registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core using the device's thing name as the client ID and to publish to any topic prefixed by that thing name or client except for one topic ending with bar:

```

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

```

### Unregistered Devices (9)

For devices not registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core using client IDs `client1` and `client2` and to publish to any topic prefixed by the client ID used to connect, except for one topic ending with bar:

```
{
}
```

```

"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iot:Connect"
        ],
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:client/client1",
            "arn:aws:iot:us-east-1:123456789012:client/client2"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "iot:Publish"
        ],
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/${iot:ClientId}/*"
        ]
    },
    {
        "Effect": "Deny",
        "Action": [
            "iot:Publish"
        ],
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/${iot:ClientId}/bar"
        ]
    }
]
}

```

## Registered Devices (10)

For devices registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core using the device's thing name as the client ID. The device can subscribe to the topic `my/topic`, but cannot publish to the `<thing-name>/bar` where `<thing-name>` is the name of the IoT thing connecting to AWS IoT Core:

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/my/topic"
            ]
        },
        {

```

```

        "Effect": "Deny",
        "Action": [
            "iot:Publish"
        ],
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/${iot:Thing.ThingName}/bar"
        ]
    }
}

```

### Unregistered Devices (10)

For devices not registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core using client ID `client1` and to subscribe to the topic `my/topic`:

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/my/topic"
            ]
        }
    ]
}

```

Thing policy variables are also replaced when a certificate or authenticated Amazon Cognito Identity is attached to a thing. The following policy grants permission to connect to AWS IoT Core with client ID `client1` and to publish and receive topic `iotmonitor/provisioning/987654321098`. It also allows the certificate holder to subscribe to this topic.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/iotmonitor/provisioning/987654321098"
            ]
        }
    ]
}

```

```
        "iot:Receive"
    ],
    "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/iotmonitor/
provisioning/987654321098"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "iot:Subscribe"
    ],
    "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/iotmonitor/
provisioning/987654321098"
    ]
}
]
```

## Policies for HTTP and WebSocket Clients

For the following operations, AWS IoT Core uses AWS IoT Core 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 Core policy attached to the Amazon Cognito Identity through the AWS IoT Core `AttachPolicy` API.

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

### Note

For other AWS IoT Core operations or for unauthenticated identities, AWS IoT Core 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 you attach to the Amazon Cognito pool role.

### HTTP

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

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${cognito-
identity.amazonaws.com:sub}"]
        }
    ]
}
```

```
        }
    ]  
}
```

To allow authenticated users, attach the preceding policy to the Amazon Cognito Identity pool role and to the Amazon Cognito Identity using the AWS IoT Core [AttachPolicy](#) API.

**Note**

When authorizing Amazon Cognito identities, AWS IoT Core considers both policies and grants the least privileges specified. An action is allowed only if both policies allow the requested action. If either policy disallows an action, that action is unauthorized.

## MQTT

To allow unauthenticated Amazon Cognito identities to publish MQTT messages over WebSocket on a topic specific to the Amazon Cognito Identity 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": ["arn:aws:iot:us-east-1:123456789012:topic/${cognito-identity.amazonaws.com:sub}"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect",
      ],
      "Resource": ["arn:aws:iot:us-east-1:123456789012:client/${cognito-identity.amazonaws.com:sub}"]
    }
  ]
}
```

To allow authenticated users, attach the preceding policy to the Amazon Cognito Identity pool role and to the Amazon Cognito Identity using the AWS IoT Core [AttachPolicy](#) API.

**Note**

When authorizing Amazon Cognito identities, AWS IoT Core considers both and grants the least privileges specified. An action is allowed only if both policies allow the requested action. If either policy disallows an action, that action is unauthorized.

## Receive Policy Examples

### Registered Devices (11)

For devices registered in AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core with a client ID that matches the thing name and to subscribe to and receive messages on the `my/topic` topic:

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

```
        "Effect": "Allow",
        "Action": [
            "iot:Connect"
        ],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
    },
    {
        "Effect": "Allow",
        "Action": [
            "iot:Subscribe"
        ],
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topicfilter/my/topic"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "iot:Receive"
        ],
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/my/topic"
        ]
    }
]
```

#### Unregistered Devices (11)

For devices not registered in AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core with client ID `client1` and to subscribe to and receive messages on one topic:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/client1"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/my/topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/my/topic"
            ]
        }
    ]
}
```

## Connect and Publish Policy Examples

For devices registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core with a client ID that matches the thing name and restricts the device to publishing on a client-ID or thing name-specific MQTT topic. For a connection to be successful, the thing name must be registered in the AWS IoT Core registry and be authenticated using an identity or principal attached to the thing:

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

For devices not registered as things in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core with client ID `client1` and restricts the device to publishing on a clientID-specific MQTT topic:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Publish"],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${iot:ClientId}"]
        },
        {
            "Effect": "Allow",
            "Action": ["iot:Connect"],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/client1"]
        }
    ]
}
```

## Certificate Policy Examples

For devices registered in AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core with a client ID that matches a thing name, and to publish to a topic whose name is equal to the `certificateId` of the certificate the device used to authenticate itself:

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

```

        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${iot:CertificateId}"]
    },
    {
        "Effect": "Allow",
        "Action": [
            "iot:Connect"
        ],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
    }
]
}

```

For devices not registered in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core with client IDs, `client1`, `client2`, and `client3` and to publish to a topic whose name is equal to the `certificateId` of the certificate the device used to authenticate 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": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        }
    ]
}

```

For devices registered in AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core with a client ID that matches the thing name, and to publish to a topic whose name is equal to the subject's `CommonName` field of the certificate the device used to authenticate itself:

```

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

```

```
        }
    ]  
}
```

**Note**

In this example, the certificate's subject common name is used as the topic identifier, with the assumption that the subject common name is unique for each registered certificate. If the certificates are shared across multiple devices, the subject common name is the same for all the devices that share this certificate, thereby allowing publish privileges to the same topic from multiple devices (not recommended).

For devices not registered in AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core with client IDs, `client1`, `client2`, and `client3` and to publish to a topic whose name is equal to the subject's `CommonName` field of the certificate the device used to authenticate itself:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Subject.CommonName}"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/client1",
        "arn:aws:iot:us-east-1:123456789012:client/client2",
        "arn:aws:iot:us-east-1:123456789012:client/client3"
      ]
    }
  ]
}
```

**Note**

In this example, the certificate's subject common name is used as the topic identifier, with the assumption that the subject common name is unique for each registered certificate. If the certificates are shared across multiple devices, the subject common name is the same for all the devices that share this certificate, thereby allowing publish privileges to the same topic from multiple devices (not recommended).

For devices registered in the AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core with a client ID that matches the thing name, and to publish to a topic whose name is prefixed with `admin/` when the certificate used to authenticate the device has its `Subject.CommonName` field set to `Administrator`:

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

```

        "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
    },
    {
        "Effect": "Allow",
        "Action": [
            "iot:Publish"
        ],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
        "Condition": {
            "StringEquals": {
                "iot:Certificate.Subject.CommonName.2": "Administrator"
            }
        }
    }
]
}

```

For devices not registered in AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core with client IDs `client1`, `client2`, and `client3` and to publish to a topic whose name 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": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
            "Condition": {
                "StringEquals": {
                    "iot:Certificate.Subject.CommonName.2": "Administrator"
                }
            }
        }
    ]
}

```

For devices registered in AWS IoT Core registry, the following policy allows a device to use its thing name to publish on a specific topic that consists of `admin/` followed by the `ThingName` when the certificate used to authenticate the device has any one of its `Subject.CommonName` fields set to `Administrator`:

```

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

```

```

        "iot:Connect"
    ],
    "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
},
{
    "Effect": "Allow",
    "Action": [
        "iot:Publish"
    ],
    "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/
${iot:Connection.Thing.ThingName}"],
    "Condition": {
        "ForAnyValue:StringEquals": {
            "iot:Certificate.Subject.CommonName.List": "Administrator"
        }
    }
}
]
}

```

For devices not registered in AWS IoT Core registry, the following policy grants permission to connect to AWS IoT Core with client IDs `client1`, `client2`, and `client3` and to publish to the topic `admin` when the certificate used to authenticate the device has any one of its `Subject.CommonName` fields set to `Administrator`:

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        },
        {
            "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 device to connect if the certificate used to authenticate with AWS IoT Core 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"]
            }
        }
    }
]
```

## Authorization with Amazon Cognito Identities

There are two types of Amazon Cognito identities: unauthenticated and authenticated. When your app supports unauthenticated Amazon Cognito identities, no authentication is performed so you do not know who the user is. For unauthenticated users, you grant permission by attaching an IAM role to an unauthenticated identity pool. You should grant access to only those resources you want available to unknown users.

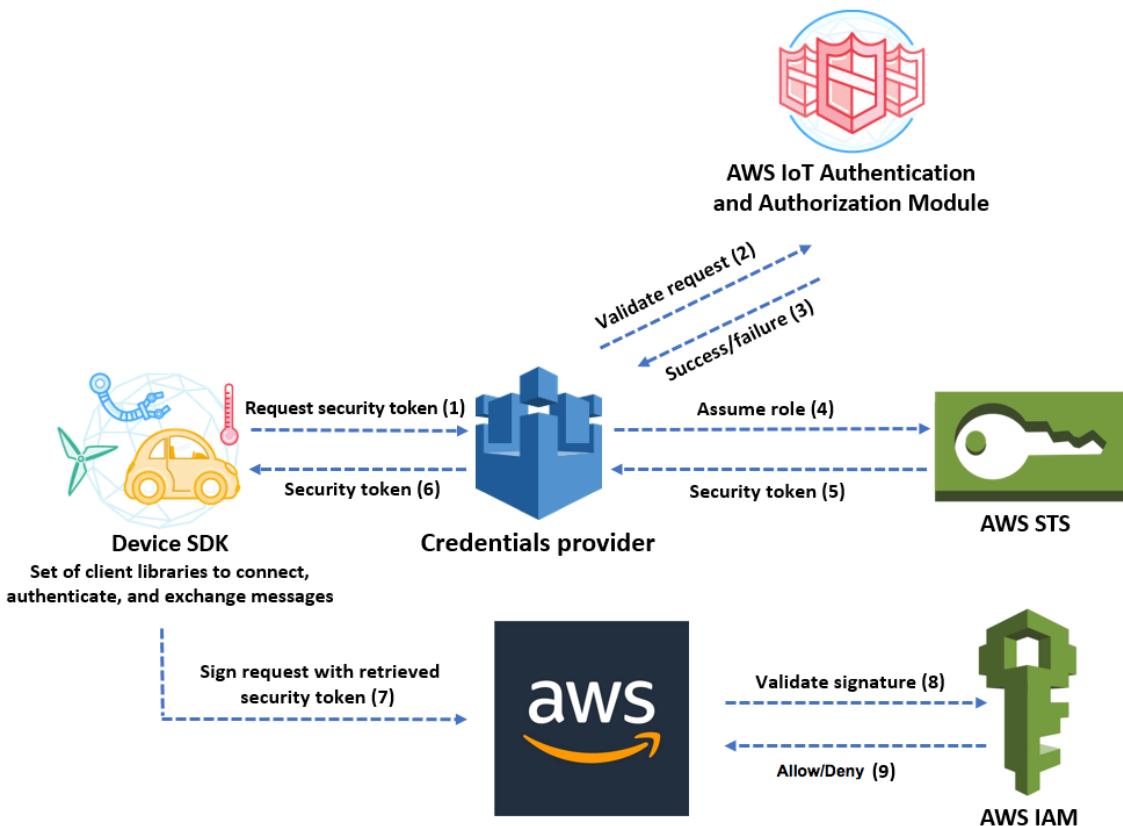
When your app supports authenticated Amazon Cognito identities, you specify a policy in two places. You attach an IAM policy to the authenticated Amazon Cognito Identity pool and you attach an AWS IoT Core policy to the Amazon Cognito Identity. You attach an IAM policy to a Amazon Cognito Identity pool using the Amazon Cognito Identity console when you create a Amazon Cognito Identity pool. To attach an AWS IoT Core policy to a Amazon Cognito Identity, you must define a Lambda function that calls `AttachPolicy`.

## Authorizing Direct Calls to AWS Services

Devices can use X.509 certificates to connect to AWS IoT Core using TLS mutual authentication protocols. Other AWS services do not support certificate-based authentication, but they can be called using AWS credentials in [AWS Signature Version 4 format](#). The [Signature Version 4 algorithm](#) normally requires the caller to have an access key ID and a secret access key. AWS IoT Core has a credentials provider that allows you to use the built-in [X.509 certificate](#) as the unique device identity to authenticate AWS requests. This eliminates the need to store an access key ID and a secret access key on your device.

The credentials provider authenticates a caller using an X.509 certificate and issues a temporary, limited-privilege security token. The token can be used to sign and authenticate any AWS request. This way of authenticating your AWS requests requires you to create and configure an [AWS Identity and Access Management \(IAM\) role](#) and attach appropriate IAM policies to the role so that the credentials provider can assume the role on your behalf. For more information about AWS IoT Core and IAM, see [Identity and Access Management for AWS IoT \(p. 174\)](#).

The following diagram illustrates the credentials provider workflow.



1. The AWS IoT Core device makes an HTTPS request to the credentials provider for a security token. The request includes the device X.509 certificate for authentication.
2. The credentials provider forwards the request to the AWS IoT Core authentication and authorization module to validate the certificate and verify that the device has permission to request the security token.
3. If the certificate is valid and has permission to request a security token, the AWS IoT Core authentication and authorization module returns success. Otherwise, it sends an exception to the device.
4. After successfully validating the certificate, the credentials provider invokes the [AWS Security Token Service \(AWS STS\)](#) to assume the IAM role that you created for it.
5. AWS STS returns a temporary, limited-privilege security token to the credentials provider.
6. The credentials provider returns the security token to the device.
7. The device uses the security token to sign an AWS request with AWS Signature Version 4.
8. The requested service invokes IAM to validate the signature and authorize the request against access policies attached to the IAM role that you created for the credentials provider.
9. If IAM validates the signature successfully and authorizes the request, the request is successful. Otherwise, IAM sends an exception.

The following section describes how to use a certificate to get a security token. It is written with the assumption that you have already [registered a device](#) and [created and activated your own certificate](#) for it.

## How to Use a Certificate to Get a Security Token

1. Configure the IAM role that the credentials provider assumes on behalf of your device. Attach the following trust policy to the role.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {"Effect": "Allow",  
         "Principal": {"Service": "credentials.iot.amazonaws.com"},  
         "Action": "sts:AssumeRole"}  
    ]  
}
```

For each AWS service that you want to call, attach an access policy to the role. The credentials provider supports the following policy variables:

- `credentials-iot:ThingName`
- `credentials-iot:ThingTypeName`
- `credentials-iot:AwsCertificateId`

When the device provides the thing name in its request to an AWS service, the credentials provider adds `credentials-iot:ThingName` and `credentials-iot:ThingTypeName` as context variables to the security token. The credentials provider provides `credentials-iot:AwsCertificateId` as a context variable even if the device doesn't provide the thing name in the request. You pass the thing name as the value of the `x-amzn-iot-thingname` HTTP request header.

These three variables work for IAM policies only, not AWS IoT Core policies.

2. Make sure that the user who performs the next step (creating a role alias) has permission to pass the newly created role to AWS IoT Core. The following policy gives both `iam:GetRole` and `iam:PassRole` permissions to an AWS user. The `iam:GetRole` permission allows the user to get information about the role that you've just created. The `iam:PassRole` permission allows the user to pass the role to another AWS service.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {"Effect": "Allow",  
         "Action": [  
             "iam:GetRole",  
             "iam:PassRole"  
         ],  
         "Resource": "arn:aws:iam::your aws account id:role/your role name"  
    ]  
}
```

3. Create an AWS IoT Core role alias. The device that is going to make direct calls to AWS services must know which role ARN to use when connecting to AWS IoT Core. Hard-coding the role ARN is not a good solution because it requires you to update the device whenever the role ARN changes. A better solution is to use the `CreateRoleAlias` API to create a role alias that points to the role ARN. If the role ARN changes, you simply update the role alias. No change is required on the device. This API takes the following parameters:

**roleAlias**

Mandatory. An arbitrary string that identifies the role alias. It serves as the primary key in the role alias data model. It contains 1-128 characters and must include only alphanumeric characters and the =, @, and - symbols. Uppercase and lowercase alphabetic characters are allowed.

**roleArn**

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

**credentialDurationInSeconds**

Optional. How long (in seconds) the credential is valid. The minimum value is 900 seconds (15 minutes). The maximum value is 3,600 seconds (60 minutes). The default value is 3,600 seconds.

For more information about this API, see [CreateRoleAlias](#).

4. Attach a policy to the device certificate. 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 to the role alias, as in the following example.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "iot:AssumeRoleWithCertificate",  
            "Resource": "arn:aws:iot:your_region:your_aws_account_id:rolealias/your_role  
alias"  
        }  
    ]  
}
```

5. Make an HTTPS request to the credentials provider to get a security token. Supply the following information:

- *Certificate*: Because this is an HTTP request over TLS mutual authentication, you must provide the certificate and the private key to your client while making the request. Use the same certificate and private key you used when you registered your certificate with AWS IoT Core.

To make sure your device is communicating with AWS IoT Core (and not a service impersonating it), see [Server Authentication](#), follow the links to download the appropriate CA certificates, and then copy them to your device.

- *RoleAlias*: The name of the role alias that you created for the credentials provider.
- *ThingName*: The thing name that you created when you registered your AWS IoT Core thing. This is passed as the value of the `x-amzn-iot-thingname` HTTP header. This value is required only if you are using thing attributes as policy variables in AWS IoT Core or IAM policies.

Run the following command in the AWS CLI to obtain the credentials provider endpoint for your AWS account. For more information about this API, see [DescribeEndpoint](#).

**aws iot describe-endpoint --endpoint-type iot:CredentialProvider**

The following JSON object is sample output of the **describe-endpoint** command. It contains the `endpointAddress` that you use to request a security token.

```
{  
    "endpointAddress": "your_aws_account_specific_prefix.credentials.iot.your  
region.amazonaws.com"
```

}

Use the endpoint to make an HTTPS request to the credentials provider to return a security token. The following example command uses curl, but you can use any HTTP client.

```
curl --cert your certificate --key your device certificate key pair -H "x-amzn-iot-thingname: your thing name" --cacert AmazonRootCA1.pem https://your endpoint /role-aliases/your role alias/credentials
```

This command returns a security token object that contains an accessKeyId, a secretAccessKey, a sessionToken, and an expiration. The following JSON object is sample output of the curl command.

```
{"credentials": {"accessKeyId": "access key", "secretAccessKey": "secret access key", "sessionToken": "session token", "expiration": "2018-01-18T09:18:06Z"}}
```

You can then use the accessKeyId, secretAccessKey, and sessionToken values to sign requests to AWS services. For an end-to-end demonstration, see [How to Eliminate the Need for Hard-Coded AWS Credentials in Devices by Using the AWS IoT Credential Provider](#) blog post on the [AWS Security Blog](#).

## Cross Account Access

AWS IoT Core 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 a customer managed IAM policy as described in [Creating IAM Policies](#), just like you would for other users and certificates in your AWS account.

For devices registered in AWS IoT Core registry, the following policy grants permission to devices connect to AWS IoT Core using a client ID that matches the device's thing name and to publish to the my/topic/<thing-name> where <thing-name> is the device's thing name:

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

For devices not registered in AWS IoT Core registry, the following policy grants permission to a device to use the thing name `client1` registered in your account's (123456789012) AWS IoT Core registry to connect to AWS IoT Core and to publish to a client ID-specific topic whose name is prefixed with `my/topic/`:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iot:Connect"  
            ],  
            "Resource": [  
                "arn:aws:iot:us-east-1:123456789012:client/client1"  
            ]  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iot:Publish"  
            ],  
            "Resource": [  
                "arn:aws:iot:us-east-1:123456789012:topic/my/topic/${iot:ClientId}"  
            ]  
        }  
    ]  
}
```

Next, follow the steps in [Creating a Role to Delegate Permissions to an IAM User](#). Enter the 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-1:567890123456:user:MyUser"  
            },  
            "Action": "sts:AssumeRole",  
        }  
    ]  
}
```

## Data Protection in AWS IoT Core

AWS IoT conforms to the AWS [shared responsibility model](#), which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that each user is given only

the permissions required to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls in AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with AWS IoT or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into AWS IoT or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the [AWS Security Blog](#).

AWS IoT devices gather data, perform some manipulation on that data, and then send that data to another web service. You might choose to store some data on your device for a short period of time. You are responsible for providing any data protection on that data at rest. When your device sends data to AWS IoT it does so over a TLS connection as discussed later in this section. AWS IoT devices can send data to any AWS service. For more information about each service's data security, see the documentation for that service. AWS IoT can be configured to write logs to CloudWatch Logs and log AWS IoT API calls to AWS CloudTrail. For more information about data security for these services, see [Authentication and Access Control for Amazon CloudWatch](#) and [Encrypting CloudTrail Log Files with AWS KMS-Managed Keys](#).

## Transport Security in AWS IoT

The AWS IoT message broker and Device Shadow service encrypt all communication with [TLS version 1.2](#). 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.

AWS IoT requires devices to send the [Server Name Indication \(SNI\) extension](#) to the Transport Layer Security (TLS) protocol and provide the complete endpoint address in the `host_name` field. The `host_name` field must contain the endpoint you are calling, and it must be:

- The `endpointAddress` returned by `aws iot describe-endpoint --endpoint-type iot:Data-ATS`  
or
- The `domainName` returned by `aws iot describe-domain-configuration --domain-configuration-name "<domain_configuration_name>"`

Connections attempted by devices without the correct `host_name` value will be refused and logged in CloudWatch.

## 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

## Data Encryption in AWS IoT

Data protection refers to protecting data while in-transit (as it travels to and from AWS IoT) and at rest (while it is stored on devices or by other AWS services). All data sent to AWS IoT is sent over an TLS connection using MQTT or HTTPS protocols, so it is secure by default in transit. AWS IoT devices collect data and then send it to other AWS services for further processing. For more information about data encryption on other AWS services, see the security documentation for that service.

FreeRTOS provides a PKCS#11 library that abstracts key storage, accessing cryptographic objects and managing sessions. It is your responsibility to use this library to encrypt data at rest on your devices. For more information, see <https://docs.aws.amazon.com/freertos/latest/userguide/security-pkcs.html>FreeRTOS Public Key Cryptography Standard (PKCS) #11 Library

## Key Management in AWS IoT

All connections to AWS IoT are done using TLS, so no client-side encryption keys are necessary for the initial TLS connection. Devices must authenticate using an X.509 certificate or an Amazon Cognito Identity. You can have AWS IoT generate a certificate for you, in which case it will generate a public/private key pair. If you are using the AWS IoT console you will be prompted to download the certificate and keys. If you are using the `create-keys-and-certificate` CLI command, the certificate and keys are returned by the CLI command. You are responsible for copying the certificate and private key onto your device and keeping it safe.

## Identity and Access Management for AWS IoT

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and

*authorized* (have permissions) to use AWS IoT resources. IAM is an AWS service that you can use with no additional charge.

#### Topics

- [Audience \(p. 175\)](#)
- [Authenticating With IAM Identities \(p. 175\)](#)
- [Managing Access Using Policies \(p. 177\)](#)
- [How AWS IoT Works with IAM \(p. 178\)](#)
- [AWS IoT Identity-Based Policy Examples \(p. 193\)](#)
- [Troubleshooting AWS IoT Identity and Access \(p. 195\)](#)

## Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work you do in AWS IoT.

**Service user** – If you use the AWS IoT service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS IoT features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in AWS IoT, see [Troubleshooting AWS IoT Identity and Access \(p. 195\)](#).

**Service administrator** – If you're in charge of AWS IoT resources at your company, you probably have full access to AWS IoT. It's your job to determine which AWS IoT features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with AWS IoT, see [How AWS IoT Works with IAM \(p. 178\)](#).

**IAM administrator** – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS IoT. To view example AWS IoT identity-based policies that you can use in IAM, see [AWS IoT Identity-Based Policy Examples \(p. 193\)](#).

## Authenticating With IAM Identities

In AWS IoT identities can be device (X.509) certificates, Amazon Cognito identities, or IAM users or groups. This topic discusses IAM identities only. For more information about the other identities that AWS IoT supports, see [Client Authentication \(p. 123\)](#).

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see [The IAM Console and Sign-in Page](#) in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication, or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the [AWS Management Console](#), use your password with your root user email or your IAM user name. You can access AWS programmatically using your root user or IAM user access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see [Signature Version 4 Signing Process](#) in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to

increase the security of your account. To learn more, see [Using Multi-Factor Authentication \(MFA\) in AWS](#) in the *IAM User Guide*.

## AWS Account Root User

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the [best practice of using the root user only to create your first IAM user](#). Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

## IAM Users and Groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see [Managing Access Keys for IAM Users](#) in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see [When to Create an IAM User \(Instead of a Role\)](#) in the *IAM User Guide*.

## IAM Roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by [switching roles](#). You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see [Using IAM Roles](#) in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an *identity provider*. For more information about federated users, see [Federated Users and Roles](#) in the *IAM User Guide*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see [How IAM Roles Differ from Resource-based Policies](#) in the *IAM User Guide*.
- **AWS service access** – A service role is an IAM role that a service assumes to perform actions in your account on your behalf. When you set up some AWS service environments, you must define a role

for the service to assume. This service role must include all the permissions that are required for the service to access the AWS resources that it needs. Service roles vary from service to service, but many allow you to choose your permissions as long as you meet the documented requirements for that service. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. You can create, modify, and delete a service role from within IAM. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see [Creating a Role to Delegate Permissions to an AWS Service](#) in the *IAM User Guide*.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances](#) in the *IAM User Guide*.

To learn whether to use IAM roles, see [When to Create an IAM Role \(Instead of a User\)](#) in the *IAM User Guide*.

## Managing Access Using Policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when an entity (root user, IAM user, or IAM role) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see [Overview of JSON Policies](#) in the *IAM User Guide*.

An IAM administrator can use policies to specify who has access to AWS resources, and what actions they can perform on those resources. Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

### Identity-Based Policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, role, or group. These policies control what actions that identity can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Creating IAM Policies](#) in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see [Choosing Between Managed Policies and Inline Policies](#) in the *IAM User Guide*.

### Resource-Based Policies

Resource-based policies are JSON policy documents that you attach to a resource such as an Amazon S3 bucket. Service administrators can use these policies to define what actions a specified principal (account

member, user, or role) can perform on that resource and under what conditions. Resource-based policies are inline policies. There are no managed resource-based policies.

## Access Control Lists (ACLs)

Access control lists (ACLs) are a type of policy that controls which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format. Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see [Access Control List \(ACL\) Overview](#) in the *Amazon Simple Storage Service Developer Guide*.

## Other Policy Types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see [Permissions Boundaries for IAM Entities](#) in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see [How SCPs Work](#) in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see [Session Policies](#) in the *IAM User Guide*.

## Multiple Policy Types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see [Policy Evaluation Logic](#) in the *IAM User Guide*.

## How AWS IoT Works with IAM

Before you use IAM to manage access to AWS IoT, you should understand which IAM features are available to use with AWS IoT. To get a high-level view of how AWS IoT and other AWS services work with IAM, see [AWS Services That Work with IAM](#) in the *IAM User Guide*.

### Topics

- [IAM Policies \(p. 179\)](#)
- [AWS IoT Identity-Based Policies \(p. 179\)](#)
- [AWS IoT Resource-Based Policies \(p. 192\)](#)
- [Authorization Based on AWS IoT Tags \(p. 192\)](#)
- [AWS IoT IAM Roles \(p. 192\)](#)

## IAM Policies

AWS IoT works with AWS IoT and IAM policies. This topic discusses IAM policies only. For more information, see [AWS IoT Core Policies \(p. 139\)](#). AWS Identity and Access Management defines a policy action for each operation defined by AWS IoT, including control plane and data plane APIs.

### IAM Managed Policies

AWS IoT provides a set of IAM managed policies you can either use as-is or as a starting point for creating custom IAM policies. These policies 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                                                                                                                                                             |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AWSIoTConfigAccess         | Allows the associated identity access to all AWS IoT configuration operations. This policy can affect data processing and storage.                                      |
| AWSIoTConfigReadOnlyAccess | Allows the associated identity to access 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.                                       |
| AWSIoTEventsFullAccess     | Allows the associated identity full access to AWS IoT events.                                                                                                           |
| AWSIoTEventsReadOnlyAccess | Allows the associated identity read only access to AWS IoT events.                                                                                                      |
| AWSIoTFullAccess           | Allows the associated identity full access to all AWS IoT configuration and messaging operations.                                                                       |
| AWSIoTLogging              | Allows the associated identity to create Amazon CloudWatch Logs groups and stream logs to the groups. This policy is attached to your CloudWatch logging role.          |
| AWSIoTOTAUUpdate           | Allows the associated identity to create AWS IoT jobs, AWS IoT code signing jobs, and to describe AWS code signer jobs.                                                 |
| AWSIoTRuleActions          | Allows the associated identity access to all AWS services supported in AWS IoT rule actions.                                                                            |
| AWSIoTThingsRegistration   | Allows the associated identity to register things in bulk using the <a href="#">StartThingRegistrationTask</a> API. This policy can affect data processing and storage. |

## AWS IoT Identity-Based Policies

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. AWS IoT supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see [IAM JSON Policy Elements Reference](#) in the *IAM User Guide*.

## Actions

The Action element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used in a policy to grant permissions to perform the associated operation.

The following table lists the IAM IoT actions, the associated AWS IoT API, and the resource the action manipulates.

| Policy Actions                | AWS IoT API               | Resources                                                                                                                                         |
|-------------------------------|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| iot:AcceptCertificateTransfer | AcceptCertificateTransfer | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                                                |
|                               |                           | <b>Note</b><br>The AWS account specified in the ARN must be the account to which the certificate is being transferred.                            |
| iot:AddThingToThingGroup      | AddThingToThingGroup      | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i><br><br>arn:aws:iot: <i>region:account-id:thing/thing-name</i>                   |
| iot:AssociateTargetsWithJob   | AssociateTargetsWithJob   |                                                                                                                                                   |
| iot:AttachPolicy              | AttachPolicy              | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i><br><br>or<br><br>arn:aws:iot: <i>region:account-id:cert/cert-id</i>             |
| iot:AttachPrincipalPolicy     | AttachPrincipalPolicy     | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                                                |
| iot:AttachSecurityProfile     | AttachSecurityProfile     | arn:aws:iot: <i>region:account-id:securityprofile/security-profile-name</i><br><br>arn:aws:iot: <i>region:account-id:dimension/dimension-name</i> |
| iot:AttachThingPrincipal      | AttachThingPrincipal      | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                                                |
| iot:CancelCertificateTransfer | CancelCertificateTransfer | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                                                |
|                               |                           | <b>Note</b><br>The AWS account specified in the ARN must be the account to which the certificate is being transferred.                            |
| iot:CancelJob                 | CancelJob                 | arn:aws:iot: <i>region:account-id:job/job-id</i>                                                                                                  |
| iot:CancelJobExecution        | CancelJobExecution        | arn:aws:iot: <i>region:account-id:job/job-id</i><br><br>arn:aws:iot: <i>region:account-id:thing/thing-name</i>                                    |
| iot:ClearDefaultAuthorizer    | ClearDefaultAuthorizer    | None                                                                                                                                              |
| iot>CreateAuthorizer          | CreateAuthorizer          | arn:aws:iot: <i>region:account-id:authorizer/authorizer-function-name</i>                                                                         |
| iot>CreateCertificateFromCsr  | CreateCertificateFromCsr  |                                                                                                                                                   |

| Policy Actions               | AWS IoT API              | Resources                                                                                                                                         |
|------------------------------|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| iot:CreateDimension          | CreateDimension          | arn:aws:iot: <i>region:account-id:dimension/dimension-name</i>                                                                                    |
| iot:CreateJob                | CreateJob                | arn:aws:iot: <i>region:account-id:job/job-id</i>                                                                                                  |
| iot:CreateKeysAndCertificate | CreateKeysAndCertificate |                                                                                                                                                   |
| iot:CreatePolicy             | CreatePolicy             | *                                                                                                                                                 |
| iot:CreatePolicyVersion      | CreatePolicyVersion      | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                                          |
|                              |                          | <b>Note</b><br>This must be an AWS IoT policy, not an IAM policy.                                                                                 |
| iot:CreateRoleAlias          | CreateRoleAlias          | (parameter: roleAlias)<br><br>arn:aws:iot: <i>region:account-id:rolealias/role-alias-name</i>                                                     |
| iot:CreateSecurityProfile    | CreateSecurityProfile    | arn:aws:iot: <i>region:account-id:securityprofile/security-profile-name</i><br><br>arn:aws:iot: <i>region:account-id:dimension/dimension-name</i> |
| iot:CreateThing              | CreateThing              | arn:aws:iot: <i>region:account-id:thing/thing-name</i>                                                                                            |
| iot:CreateThingGroup         | CreateThingGroup         | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i><br><br>for group being created and for parent group, if used                    |
| iot:CreateThingType          | CreateThingType          | arn:aws:iot: <i>region:account-id:thingtype/thing-type-name</i>                                                                                   |
| iot:CreateTopicRule          | CreateTopicRule          | arn:aws:iot: <i>region:account-id:rule/rule-name</i>                                                                                              |
| iot:DeleteAuthorizer         | DeleteAuthorizer         | arn:aws:iot: <i>region:account-id:authorizer/authorizer-name</i>                                                                                  |
| iot:DeleteCACertificate      | DeleteCACertificate      | arn:aws:iot: <i>region:account-id:cacert/cert-id</i>                                                                                              |
| iot:DeleteCertificate        | DeleteCertificate        | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                                                |
| iot:DeleteDimension          | DeleteDimension          | arn:aws:iot: <i>region:account-id:dimension/dimension-name</i>                                                                                    |
| iot:DeleteJob                | DeleteJob                | arn:aws:iot: <i>region:account-id:job/job-id</i>                                                                                                  |
| iot:DeleteJobExecution       | DeleteJobExecution       | arn:aws:iot: <i>region:account-id:job/job-id</i><br><br>arn:aws:iot: <i>region:account-id:thing/thing-name</i>                                    |
| iot:DeletePolicy             | DeletePolicy             | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                                          |
| iot:DeletePolicyVersion      | DeletePolicyVersion      | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                                          |
| iot:DeleteRegistrationCode   | DeleteRegistrationCode   |                                                                                                                                                   |

| Policy Actions                    | AWS IoT API                   | Resources                                                                                                                                                       |
|-----------------------------------|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| iot:DeleteRoleAlias               | DeleteRoleAlias               | arn:aws:iot: <i>region:account-id</i> :rolealias/ <i>role-alias-name</i>                                                                                        |
| iot:DeleteSecurityProfile         | DeleteSecurityProfile         | arn:aws:iot: <i>region:account-id</i> :securityprofile/ <i>security-profile-name</i><br>arn:aws:iot: <i>region:account-id</i> :dimension/ <i>dimension-name</i> |
| iot:DeleteThing                   | DeleteThing                   | arn:aws:iot: <i>region:account-id</i> :thing/ <i>thing-name</i>                                                                                                 |
| iot:DeleteThingGroup              | DeleteThingGroup              | arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i>                                                                                      |
| iot:DeleteThingType               | DeleteThingType               | arn:aws:iot: <i>region:account-id</i> :thingtype/ <i>thing-type-name</i>                                                                                        |
| iot:DeleteTopicRule               | DeleteTopicRule               | arn:aws:iot: <i>region:account-id</i> :rule/ <i>rule-name</i>                                                                                                   |
| iot:DeleteV2LoggingLevel          | DeleteV2LoggingLevel          | arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i>                                                                                      |
| iot:DeprecateThingType            | DeprecateThingType            | arn:aws:iot: <i>region:account-id</i> :thingtype/ <i>thing-type-name</i>                                                                                        |
| iot:DescribeAuthorizer            | DescribeAuthorizer            | arn:aws:iot: <i>region:account-id</i> :authorizer/ <i>authorizer-function-name</i><br>(parameter: authorizerName)<br>none                                       |
| iot:DescribeCACertificate         | DescribeCACertificate         | arn:aws:iot: <i>region:account-id</i> :cacert/ <i>cert-id</i>                                                                                                   |
| iot:DescribeCertificate           | DescribeCertificate           | arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>                                                                                                     |
| iot:DescribeDefaultAuthorizer     | DescribeDefaultAuthorizer     | None                                                                                                                                                            |
| iot:DescribeEndpoint              | DescribeEndpoint              | *                                                                                                                                                               |
| iot:DescribeEventConfiguration    | DescribeEventConfiguration    | None                                                                                                                                                            |
| iot:DescribeIndex                 | DescribeIndex                 | arn:aws:iot: <i>region:account-id</i> :index/ <i>index-name</i>                                                                                                 |
| iot:DescribeJob                   | DescribeJob                   | arn:aws:iot: <i>region:account-id</i> :job/ <i>job-id</i>                                                                                                       |
| iot:DescribeJobExecution          | DescribeJobExecution          | None                                                                                                                                                            |
| iot:DescribeRoleAlias             | DescribeRoleAlias             | arn:aws:iot: <i>region:account-id</i> :rolealias/ <i>role-alias-name</i>                                                                                        |
| iot:DescribeThing                 | DescribeThing                 | arn:aws:iot: <i>region:account-id</i> :thing/ <i>thing-name</i>                                                                                                 |
| iot:DescribeThingGroup            | DescribeThingGroup            | arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i>                                                                                      |
| iot:DescribeThingRegistrationTask | DescribeThingRegistrationTask | None                                                                                                                                                            |
| iot:DescribeThingType             | DescribeThingType             | arn:aws:iot: <i>region:account-id</i> :thingtype/ <i>thing-type-name</i>                                                                                        |

| Policy Actions                | AWS IoT API               | Resources                                                                                                                                     |
|-------------------------------|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| iot:DetachPolicy              | DetachPolicy              | arn:aws:iot: <i>region:account-id:cert/cert-id</i><br>or<br>arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i>                 |
| iot:DetachPrincipalPolicy     | DetachPrincipalPolicy     | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                                            |
| iot:DetachSecurityProfile     | DetachSecurityProfile     | arn:aws:iot: <i>region:account-id:securityprofile/security-profile-name</i><br>arn:aws:iot: <i>region:account-id:dimension/dimension-name</i> |
| iot:DetachThingPrincipal      | DetachThingPrincipal      | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                                            |
| iot:DisableTopicRule          | DisableTopicRule          | arn:aws:iot: <i>region:account-id:rule/rule-name</i>                                                                                          |
| iot:EnableTopicRule           | EnableTopicRule           | arn:aws:iot: <i>region:account-id:rule/rule-name</i>                                                                                          |
| iot:GetEffectivePolicies      | GetEffectivePolicies      | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                                            |
| iot:GetIndexingConfig         | GetIndexingConfiguration  | *                                                                                                                                             |
| iot:GetJobDocument            | GetJobDocument            | arn:aws:iot: <i>region:account-id:job/job-id</i>                                                                                              |
| iot:GetLoggingOptions         | GetLoggingOptions         | *                                                                                                                                             |
| iot:GetPolicy                 | GetPolicy                 | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                                      |
| iot:GetPolicyVersion          | GetPolicyVersion          | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                                      |
| iot:GetRegistrationCode       | GetRegistrationCode       | *                                                                                                                                             |
| iot:GetTopicRule              | GetTopicRule              | arn:aws:iot: <i>region:account-id:rule/rule-name</i>                                                                                          |
| iot>ListAttachedPolicies      | ListAttachedPolicies      | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i><br>or<br>arn:aws:iot: <i>region:account-id:cert/cert-id</i>                 |
| iot>ListAuthorizers           | ListAuthorizers           | None                                                                                                                                          |
| iot>ListCACertificates        | ListCACertificates        | *                                                                                                                                             |
| iot>ListCertificates          | ListCertificates          | *                                                                                                                                             |
| iot>ListCertificatesByCA      | ListCertificatesByCA      | *                                                                                                                                             |
| iot>ListIndices               | ListIndices               | None                                                                                                                                          |
| iot>ListJobExecutions         | ListJobExecutions         | None                                                                                                                                          |
| iot>ListJobExecutionsForThing | ListJobExecutionsForThing | None                                                                                                                                          |

| Policy Actions                 | AWS IoT API                | Resources                                                                                                                   |
|--------------------------------|----------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| iot:ListJobs                   | ListJobs                   | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i><br>if thingGroupName parameter used                       |
| iot:ListOutgoingCertificates   | ListCertificates           |                                                                                                                             |
| iot:ListPolicies               | ListPolicies               | *                                                                                                                           |
| iot:ListPolicyPrincipals       | ListPolicyPrincipals       | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                    |
| iot:ListPolicyVersions         | ListPolicyVersions         | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                    |
| iot:ListPrincipalPolicies      | ListPrincipalPolicies      | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                          |
| iot:ListPrincipalThings        | ListPrincipalThings        | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                          |
| iot:ListRoleAliases            | ListRoleAliases            | None                                                                                                                        |
| iot:ListTargetsForPolicy       | ListTargetsForPolicy       | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                    |
| iot:ListThingGroups            | ListThingGroups            | None                                                                                                                        |
| iot:ListThingGroupsForThing    | ListThingGroupsForThing    | arn:aws:iot: <i>region:account-id:thing/thing-name</i>                                                                      |
| iot:ListThingPrincipals        | ListThingPrincipals        | arn:aws:iot: <i>region:account-id:thing/thing-name</i>                                                                      |
| iot:ListThingRegistrationTasks | ListThingRegistrationTasks | Task Reports                                                                                                                |
| iot:ListThingRegistrations     | ListThingRegistrations     | Tasks                                                                                                                       |
| iot:ListThingTypes             | ListThingTypes             | *                                                                                                                           |
| iot:ListThings                 | ListThings                 | *                                                                                                                           |
| iot:ListThingsInThingGroup     | ListThingsInThingGroup     | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i>                                                           |
| iot:ListTopicRules             | ListTopicRules             | *                                                                                                                           |
| iot:ListV2LoggingLevel         | ListV2LoggingLevels        | None                                                                                                                        |
| iot:RegisterCACertificate      | RegisterCACertificate      | *                                                                                                                           |
| iot:RegisterCertificate        | RegisterCertificate        | *                                                                                                                           |
| iot:RegisterThing              | RegisterThing              | None                                                                                                                        |
| iot:RejectCertificateTransfer  | RejectCertificateTransfer  | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                          |
| iot:RemoveThingFromThingGroup  | RemoveThingFromThingGroup  | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i><br>arn:aws:iot: <i>region:account-id:thing/thing-name</i> |
| iot:ReplaceTopicRule           | ReplaceTopicRule           | arn:aws:iot: <i>region:account-id:rule/rule-name</i>                                                                        |
| iot:SearchIndex                | SearchIndex                | arn:aws:iot: <i>region:account-id:index/index-id</i>                                                                        |
| iot:SetDefaultAuthorizer       | SetDefaultAuthorizer       | arn:aws:iot: <i>region:account-id:authorizer/authorizer-function-name</i>                                                   |

| Policy Actions                  | AWS IoT API                 | Resources                                                                                                                                 |
|---------------------------------|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| iot:SetDefaultPolicy            | SetDefaultPolicyVersion     | arn:aws:iot:<region>:<account-id>:policy/<policy-name>                                                                                    |
| iot:SetLoggingOptions           | SetLoggingOptions           | arn:aws:iot:<region>:<account-id>:role/<role-name>                                                                                        |
| iot:SetV2LoggingLevel           | SetV2LoggingLevel           | arn:aws:iot:<region>:<account-id>:thinggroup/<thing-group-name>                                                                           |
| iot:SetV2LoggingOptions         | SetV2LoggingOptions         | arn:aws:iot:<region>:<account-id>:role/<role-name>                                                                                        |
| iot:StartThingRegistrationTask  | StartThingRegistrationTask  | No Task                                                                                                                                   |
| iot:StopThingRegistrationTask   | StopThingRegistrationTask   | No Task                                                                                                                                   |
| iot:TestAuthorization           | TestAuthorization           | arn:aws:iot:<region>:<account-id>:cert/<cert-id>                                                                                          |
| iot:TestInvokeAuthorizer        | TestInvokeAuthorizer        | None                                                                                                                                      |
| iot:TransferCertificate         | TransferCertificate         | arn:aws:iot:<region>:<account-id>:cert/<cert-id>                                                                                          |
| iot:UpdateAuthorizer            | UpdateAuthorizer            | arn:aws:iot:<region>:<account-id>:authorizerfunction/<authorizer-function-name>                                                           |
| iot:UpdateCACertificate         | UpdateCACertificate         | arn:aws:iot:<region>:<account-id>:cacert/<cert-id>                                                                                        |
| iot:UpdateCertificate           | UpdateCertificate           | arn:aws:iot:<region>:<account-id>:cert/<cert-id>                                                                                          |
| iot:UpdateDimension             | UpdateDimension             | arn:aws:iot:<region>:<account-id>:dimension/<dimension-name>                                                                              |
| iot:UpdateEventConfiguration    | UpdateEventConfiguration    | No Task                                                                                                                                   |
| iot:UpdateIndexingConfiguration | UpdateIndexingConfiguration | No Task                                                                                                                                   |
| iot:UpdateRoleAlias             | UpdateRoleAlias             | arn:aws:iot:<region>:<account-id>:rolealias/<role-alias-name>                                                                             |
| iot:UpdateSecurityProfile       | UpdateSecurityProfile       | arn:aws:iot:<region>:<account-id>:securityprofile/<security-profile-name><br>arn:aws:iot:<region>:<account-id>:dimension/<dimension-name> |
| iot:UpdateThing                 | UpdateThing                 | arn:aws:iot:<region>:<account-id>:thing/<thing-name>                                                                                      |
| iot:UpdateThingGroup            | UpdateThingGroup            | arn:aws:iot:<region>:<account-id>:thinggroup/<thing-group-name>                                                                           |
| iot:UpdateThingGroups           | UpdateThingGroups           | arn:aws:iot:<region>:<account-id>:thing/<thing-name>                                                                                      |

Policy actions in AWS IoT use the following prefix before the action: `iot:`. For example, to grant someone permission to list all IoT things registered in their AWS account with the `ListThings` API, you include the `iot:ListThings` action in their policy. Policy statements must include either an `Action` or `NotAction` element. AWS IoT defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows:

```
"Action": [
    "ec2:action1",
```

"ec2:action2"

You can specify multiple actions using wildcards (\*). For example, to specify all actions that begin with the word `Describe`, include the following action:

"Action": "iot:Describe\*"

To see a list of AWS IoT actions, see [Actions Defined by AWS IoT](#) in the *IAM User Guide*.

## Resources

The `Resource` element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. You specify a resource using an ARN or using the wildcard (\*) to indicate that the statement applies to all resources.

### AWS IoT Resources

| Policy Actions                | AWS IoT API                  | Resources                                                                                                                     |
|-------------------------------|------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| iot:AcceptCertificateTransfer | TransferCertificateTransfers | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                            |
|                               |                              | <b>Note</b><br>The AWS account specified in the ARN must be the account to which the certificate is being transferred.        |
| iot:AddThingToThingGroup      | AddThingToThingGroup         | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i><br>arn:aws:iot: <i>region:account-id:thing/thing-name</i>   |
| iot:AssociateTargetsWithJob   | AssociateTargetsWithJob      |                                                                                                                               |
| iot:AttachPolicy              | AttachPolicy                 | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i><br>or<br>arn:aws:iot: <i>region:account-id:cert/cert-id</i> |
| iot:AttachPrincipalPolicy     | AttachPrincipalPolicy        | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                            |
| iot:AttachThingPrincipal      | AttachThingPrincipal         | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                            |
| iot:CancelCertificateTransfer | CancelCertificateTransfers   | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                            |
|                               |                              | <b>Note</b><br>The AWS account specified in the ARN must be the account to which the certificate is being transferred.        |
| iot:CancelJob                 | CancelJob                    | arn:aws:iot: <i>region:account-id:job/job-id</i>                                                                              |
| iot:CancelJobExecution        | CancelJobExecution           | arn:aws:iot: <i>region:account-id:job/job-id</i><br>arn:aws:iot: <i>region:account-id:thing/thing-name</i>                    |
| iot:ClearDefaultAuthorization | ClearDefaultAuthorization    | None                                                                                                                          |
| iot>CreateAuthorizer          | CreateAuthorizer             | arn:aws:iot: <i>region:account-id:authorizer/authorizer-function-name</i>                                                     |
| iot>CreateCertificateFromCsr  | CreateCertificateFromCsr     |                                                                                                                               |

| Policy Actions                | AWS IoT API              | Resources                                                                                                                      |
|-------------------------------|--------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| iot:CreateJob                 | CreateJob                | arn:aws:iot: <i>region:account-id:job/job-id</i>                                                                               |
| iot:CreateKeysAndCertificates | CreateKeysAndCertificate |                                                                                                                                |
| iot:CreatePolicy              | CreatePolicy             | *                                                                                                                              |
| CreatePolicyVersion           | iot:CreatePolicyVersion  | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                       |
|                               |                          | <b>Note</b><br>This must be an AWS IoT policy, not an IAM policy.                                                              |
| iot:CreateRoleAlias           | CreateRoleAlias          | (parameter: roleAlias)<br><br>arn:aws:iot: <i>region:account-id:rolealias/role-alias-name</i>                                  |
| iot:CreateThing               | CreateThing              | arn:aws:iot: <i>region:account-id:thing/thing-name</i>                                                                         |
| iot:CreateThingGroup          | CreateThingGroup         | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i><br><br>for group being created and for parent group, if used |
| iot:CreateThingType           | CreateThingType          | arn:aws:iot: <i>region:account-id:thingtype/thing-type-name</i>                                                                |
| iot:CreateTopicRule           | CreateTopicRule          | arn:aws:iot: <i>region:account-id:rule/rule-name</i>                                                                           |
| iot:DeleteAuthorizer          | DeleteAuthorizer         | arn:aws:iot: <i>region:account-id:authorizer/authorizer-name</i>                                                               |
| iot:DeleteCACertificate       | DeleteCACertificate      | arn:aws:iot: <i>region:account-id:cacert/cert-id</i>                                                                           |
| iot:DeleteCertificate         | DeleteCertificate        | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                             |
| iot:DeleteJob                 | DeleteJob                | arn:aws:iot: <i>region:account-id:job/job-id</i>                                                                               |
| iot:DeleteJobExecution        | DeleteJobExecution       | arn:aws:iot: <i>region:account-id:job/job-id</i><br><br>arn:aws:iot: <i>region:account-id:thing/thing-name</i>                 |
| iot:DeletePolicy              | DeletePolicy             | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                       |
| iot:DeletePolicyVersion       | DeletePolicyVersion      | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                       |
| iot:DeleteRegistrationCode    | DeleteRegistrationCode   |                                                                                                                                |
| iot:DeleteRoleAlias           | DeleteRoleAlias          | arn:aws:iot: <i>region:account-id:rolealias/role-alias-name</i>                                                                |
| iot:DeleteThing               | DeleteThing              | arn:aws:iot: <i>region:account-id:thing/thing-name</i>                                                                         |
| iot:DeleteThingGroup          | DeleteThingGroup         | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i>                                                              |
| iot:DeleteThingType           | DeleteThingType          | arn:aws:iot: <i>region:account-id:thingtype/thing-type-name</i>                                                                |
| iot:DeleteTopicRule           | DeleteTopicRule          | arn:aws:iot: <i>region:account-id:rule/rule-name</i>                                                                           |

| Policy Actions                    | AWS IoT API                | Resources                                                                                                                             |
|-----------------------------------|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| iot:DeleteV2LoggingLevel          | DeleteV2LoggingLevel       | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i>                                                                     |
| iot:DeprecateThingType            | DeprecateThingType         | arn:aws:iot: <i>region:account-id:thingtype/thing-type-name</i>                                                                       |
| iot:DescribeAuthorizedFunction    | DescribeAuthorizer         | arn:aws:iot: <i>region:account-id:authorizer/authorizer-function-name</i><br><br>(parameter: authorizerName)<br>none                  |
| iot:DescribeCACertificate         | DescribeCACertificate      | arn:aws:iot: <i>region:account-id:cacert/cert-id</i>                                                                                  |
| iot:DescribeCertificate           | DescribeCertificate        | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                                    |
| iot:DescribeDefaultAuthorizer     | DescribeDefaultAuthorizer  |                                                                                                                                       |
| iot:DescribeEndpoint              | DescribeEndpoint           | *                                                                                                                                     |
| iot:DescribeEventConfiguration    | DescribeEventConfiguration |                                                                                                                                       |
| iot:DescribeIndex                 | DescribeIndex              | arn:aws:iot: <i>region:account-id:index/index-name</i>                                                                                |
| iot:DescribeJob                   | DescribeJob                | arn:aws:iot: <i>region:account-id:job/job-id</i>                                                                                      |
| iot:DescribeJobExecution          | DescribeJobExecution       | None                                                                                                                                  |
| iot:DescribeRoleAlias             | DescribeRoleAlias          | arn:aws:iot: <i>region:account-id:rolealias/role-alias-name</i>                                                                       |
| iot:DescribeThing                 | DescribeThing              | arn:aws:iot: <i>region:account-id:thing/thing-name</i>                                                                                |
| iot:DescribeThingGroup            | DescribeThingGroup         | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i>                                                                     |
| iot:DescribeThingRegistrationTask | DescribeTaskRegistration   | Task                                                                                                                                  |
| iot:DescribeThingType             | DescribeThingType          | arn:aws:iot: <i>region:account-id:thingtype/thing-type-name</i>                                                                       |
| iot:DetachPolicy                  | DetachPolicy               | arn:aws:iot: <i>region:account-id:cert/cert-id</i><br><br>or<br><br>arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i> |
| iot:DetachPrincipalPolicy         | DetachPrincipalPolicy      | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                                    |
| iot:DetachThingPrincipal          | DetachThingPrincipal       | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                                    |
| iot:DisableTopicRule              | DisableTopicRule           | arn:aws:iot: <i>region:account-id:rule/rule-name</i>                                                                                  |
| iot:EnableTopicRule               | EnableTopicRule            | arn:aws:iot: <i>region:account-id:rule/rule-name</i>                                                                                  |
| iot:GetEffectivePolicies          | GetEffectivePolicies       | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                                    |
| iot:GetIndexingConfiguration      | GetIndexingConfiguration   |                                                                                                                                       |

| Policy Actions                | AWS IoT API               | Resources                                                                                                                     |
|-------------------------------|---------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| iot:GetJobDocument            | GetJobDocument            | arn:aws:iot: <i>region:account-id:job/job-id</i>                                                                              |
| iot:GetLoggingOptions         | GetLoggingOptions         | *                                                                                                                             |
| iot:GetPolicy                 | GetPolicy                 | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                      |
| iot:GetPolicyVersion          | GetPolicyVersion          | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                      |
| iot:GetRegistrationCode       | GetRegistrationCode       | *                                                                                                                             |
| iot:GetTopicRule              | GetTopicRule              | arn:aws:iot: <i>region:account-id:rule/rule-name</i>                                                                          |
| iot>ListAttachedPolicies      | ListAttachedPolicies      | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i><br>or<br>arn:aws:iot: <i>region:account-id:cert/cert-id</i> |
| iot>ListAuthorizers           | ListAuthorizers           | None                                                                                                                          |
| iot>ListCACertificates        | ListCACertificates        | *                                                                                                                             |
| iot>ListCertificates          | ListCertificates          | *                                                                                                                             |
| iot>ListCertificatesByCA      | ListCertificatesByCA      | *                                                                                                                             |
| iot>ListIndices               | ListIndices               | None                                                                                                                          |
| iot>ListJobExecutionsForJob   | ListJobExecutionsForJob   | None                                                                                                                          |
| iot>ListJobExecutionsForThing | ListJobExecutionsForThing | None                                                                                                                          |
| iot>ListJobs                  | ListJobs                  | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i><br>if thingGroupName parameter used                         |
| iot>ListOutgoingCertificates  | ListOutgoingCertificates  |                                                                                                                               |
| iot>ListPolicies              | ListPolicies              | *                                                                                                                             |
| iot>ListPolicyPrincipals      | ListPolicyPrincipals      | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                      |
| iot>ListPolicyVersions        | ListPolicyVersions        | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                      |
| iot>ListPrincipalPolicies     | ListPrincipalPolicies     | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                            |
| iot>ListPrincipalThings       | ListPrincipalThings       | arn:aws:iot: <i>region:account-id:cert/cert-id</i>                                                                            |
| iot>ListRoleAliases           | ListRoleAliases           | None                                                                                                                          |
| iot>ListTargetsForPolicy      | ListTargetsForPolicy      | arn:aws:iot: <i>region:account-id:policy/policy-name</i>                                                                      |
| iot>ListThingGroups           | ListThingGroups           | None                                                                                                                          |
| iot>ListThingGroupsForThing   | ListThingGroupsForThing   | arn:aws:iot: <i>region:account-id:thing/thing-name</i>                                                                        |
| iot>ListThingPrincipals       | ListThingPrincipals       | arn:aws:iot: <i>region:account-id:thing/thing-name</i>                                                                        |
| iot>ListThingRegistrations    | ListThingRegistrations    | None                                                                                                                          |
|                               |                           | Reports                                                                                                                       |

| Policy Actions                 | AWS IoT API                | Resources                                                                                                                   |
|--------------------------------|----------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| iot:ListThingRegistrationTasks | ListThingRegistrationTasks |                                                                                                                             |
| iot:ListThingTypes             | ListThingTypes             | *                                                                                                                           |
| iot:ListThings                 | ListThings                 | *                                                                                                                           |
| iot:ListThingsInThingGroup     | ListThingsInThingGroup     | arn:aws:iot:region:account-id:thinggroup/ <i>thing-group-name</i>                                                           |
| iot:ListTopicRules             | ListTopicRules             | *                                                                                                                           |
| iot:ListV2LoggingLevels        | ListV2LoggingLevels        | None                                                                                                                        |
| iot:RegisterCACertificate      | RegisterCACertificate      | *                                                                                                                           |
| iot:RegisterCertificate        | RegisterCertificate        | *                                                                                                                           |
| iot:RegisterThing              | RegisterThing              | None                                                                                                                        |
| iot:RejectCertificateTransfer  | RejectCertificateTransfer  | arn:aws:iot:region:account-id:cert/ <i>cert-id</i>                                                                          |
| iot:RemoveThingFromThingGroup  | RemoveThingFromThingGroup  | arn:aws:iot:region:account-id:thinggroup/ <i>thing-group-name</i><br>arn:aws:iot:region:account-id:thing/ <i>thing-name</i> |
| iot:ReplaceTopicRule           | ReplaceTopicRule           | arn:aws:iot:region:account-id:rule/ <i>rule-name</i>                                                                        |
| iot:SearchIndex                | SearchIndex                | arn:aws:iot:region:account-id:index/ <i>index-id</i>                                                                        |
| iot:SetDefaultAuthorizer       | SetDefaultAuthorizer       | arn:aws:iot:region:account-id:authorizer/ <i>authorizer-function-name</i>                                                   |
| iot:SetDefaultPolicyVersion    | SetDefaultPolicyVersion    | arn:aws:iot:region:account-id:policy/ <i>policy-name</i>                                                                    |
| iot:SetLoggingOptions          | SetLoggingOptions          | arn:aws:iot:region:account-id:role/ <i>role-name</i>                                                                        |
| iot:SetV2LoggingLevel          | SetV2LoggingLevel          | arn:aws:iot:region:account-id:thinggroup/ <i>thing-group-name</i>                                                           |
| iot:SetV2LoggingOptions        | SetV2LoggingOptions        | arn:aws:iot:region:account-id:role/ <i>role-name</i>                                                                        |
| iot:StartThingRegistrationTask | StartThingRegistrationTask |                                                                                                                             |
| iot:StopThingRegistrationTask  | StopThingRegistrationTask  |                                                                                                                             |
| iot:TestAuthorization          | TestAuthorization          | arn:aws:iot:region:account-id:cert/ <i>cert-id</i>                                                                          |
| iot:TestInvokeAuthorizer       | TestInvokeAuthorizer       | None                                                                                                                        |
| iot:TransferCertificate        | TransferCertificate        | arn:aws:iot:region:account-id:cert/ <i>cert-id</i>                                                                          |
| iot:UpdateAuthorizer           | UpdateAuthorizer           | arn:aws:iot:region:account-id:authorizerfunction/ <i>authorizer-function-name</i>                                           |
| iot:UpdateCACertificate        | UpdateCACertificate        | arn:aws:iot:region:account-id:cacert/ <i>cert-id</i>                                                                        |
| iot:UpdateCertificate          | UpdateCertificate          | arn:aws:iot:region:account-id:cert/ <i>cert-id</i>                                                                          |
| iot:UpdateEventConfigurations  | UpdateEventConfigurations  | None                                                                                                                        |

| Policy Actions                  | AWS IoT API                 | Resources                                                         |
|---------------------------------|-----------------------------|-------------------------------------------------------------------|
| iot:UpdateIndexingConfiguration | UpdateIndexingConfiguration |                                                                   |
| iot:UpdateRoleAlias             | UpdateRoleAlias             | arn:aws:iot: <i>region:account-id:rolealias/role-alias-name</i>   |
| iot:UpdateThing                 | UpdateThing                 | arn:aws:iot: <i>region:account-id:thing/thing-name</i>            |
| iot:UpdateThingGroup            | UpdateThingGroup            | arn:aws:iot: <i>region:account-id:thinggroup/thing-group-name</i> |
| iot:UpdateThingGroups           | UpdateThingGroups           | arn:aws:iot: <i>region:account-id:thing/thing-name</i>            |

For more information about the format of ARNs, see [Amazon Resource Names \(ARNs\) and AWS Service Namespaces](#).

Some AWS IoT actions, such as those for creating resources, cannot be performed on a specific resource. In those cases, you must use the wildcard (\*).

"Resource": "\*"

To see a list of AWS IoT resource types and their ARNs, see [Resources Defined by AWS IoT](#) in the *IAM User Guide*. To learn with which actions you can specify the ARN of each resource, see [Actions Defined by AWS IoT](#).

## Condition Keys

The Condition element (or Condition *block*) lets you specify conditions in which a statement is in effect. The Condition element is optional. You can build conditional expressions that use [condition operators](#), such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple Condition elements in a statement, or multiple keys in a single Condition element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see [IAM Policy Elements: Variables and Tags](#) in the *IAM User Guide*.

AWS IoT defines its own set of condition keys and also supports using some global condition keys. To see all AWS global condition keys, see [AWS Global Condition Context Keys](#) in the *IAM User Guide*.

## AWS IoT Condition Keys

| AWS IoT Condition Keys        | Description                                                              | Type   |
|-------------------------------|--------------------------------------------------------------------------|--------|
| aws:RequestTag/\${<tag-key>}  | A tag key that is present in the request that the user makes to AWS IoT. | String |
| aws:ResourceTag/\${<tag-key>} | The tag key component of a tag attached to an AWS IoT resource.          | String |

| AWS IoT Condition Keys | Description                                                                    | Type   |
|------------------------|--------------------------------------------------------------------------------|--------|
| aws:TagKeys            | The list of all the tag key names associated with the resource in the request. | String |

To see a list of AWS IoT condition keys, see [Condition Keys for AWS IoT](#) in the *IAM User Guide*. To learn with which actions and resources you can use a condition key, see [Actions Defined by AWS IoT](#).

## Examples

To view examples of AWS IoT identity-based policies, see [AWS IoT Identity-Based Policy Examples \(p. 193\)](#).

## AWS IoT Resource-Based Policies

Resource-based policies are JSON policy documents that specify what actions a specified principal can perform on the AWS IoT resource and under what conditions.

AWS IoT does not support IAM resource-based policies. It does, however, support AWS IoT resource-based policies.

## Authorization Based on AWS IoT Tags

You can attach tags to AWS IoT resources or pass tags in a request to AWS IoT. To control access based on tags, you provide tag information in the [condition element](#) of a policy using the `iot:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys. For more information, see [Using Tags with IAM Policies \(p. 115\)](#). For more information about tagging AWS IoT resources, see [Tagging Your AWS IoT Resources \(p. 114\)](#).

To view an example identity-based policy for limiting access to a resource based on the tags on that resource, see [Viewing AWS IoT Resources Based on Tags \(p. 194\)](#).

## AWS IoT IAM Roles

An [IAM role](#) is an entity within your AWS account that has specific permissions.

## Using Temporary Credentials with AWS IoT

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as [AssumeRole](#) or [GetFederationToken](#).

AWS IoT supports using temporary credentials.

## Service-Linked Roles

[Service-linked roles](#) allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view but not edit the permissions for service-linked roles.

AWS IoT does not support service-linked roles.

## Service Roles

This feature allows a service to assume a [service role](#) on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your IAM account and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, doing so might break the functionality of the service.

AWS IoT does not support service roles.

## AWS IoT Identity-Based Policy Examples

By default, IAM users and roles don't have permission to create or modify AWS IoT resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see [Creating Policies on the JSON Tab](#) in the *IAM User Guide*.

### Topics

- [Policy Best Practices \(p. 193\)](#)
- [Using the AWS IoT Console \(p. 193\)](#)
- [Allow Users to View Their Own Permissions \(p. 194\)](#)
- [Viewing AWS IoT Resources Based on Tags \(p. 194\)](#)

## Policy Best Practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete AWS IoT resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get Started Using AWS Managed Policies** – To start using AWS IoT quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see [Get Started Using Permissions With AWS Managed Policies](#) in the *IAM User Guide*.
- **Grant Least Privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see [Grant Least Privilege](#) in the *IAM User Guide*.
- **Enable MFA for Sensitive Operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see [Using Multi-Factor Authentication \(MFA\) in AWS](#) in the *IAM User Guide*.
- **Use Policy Conditions for Extra Security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see [IAM JSON Policy Elements: Condition](#) in the *IAM User Guide*.

## Using the AWS IoT Console

To access the AWS IoT console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the AWS IoT resources in your AWS account. If you create an

identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (IAM users or roles) with that policy.

To ensure that those entities can still use the AWS IoT console, also attach the following AWS managed policy to the entities: `AWSIoTFullAccess`. For more information, see [Adding Permissions to a User](#) in the *IAM User Guide*.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that you're trying to perform.

## Allow Users to View Their Own Permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam>ListGroupsForUser",
                "iam>ListAttachedUserPolicies",
                "iam>ListUserPolicies",
                "iam GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam>ListAttachedGroupPolicies",
                "iam>ListGroupPolicies",
                "iam>ListPolicyVersions",
                "iam>ListPolicies",
                "iam>ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

## Viewing AWS IoT Resources Based on Tags

You can use conditions in your identity-based policy to control access to AWS IoT resources based on tags. This example shows how you might create a policy that allows viewing a thing. However, permission is granted only if the thing tag `Owner` has the value of that user's user name. This policy also grants the permissions necessary to complete this action on the console.

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

```

```
        "Sid": "ListBillingGroupsInConsole",
        "Effect": "Allow",
        "Action": "iot>ListBillingGroups",
        "Resource": "*"
    },
{
    "Sid": "ViewBillingGroupsIfOwner",
    "Effect": "Allow",
    "Action": "iot>DescribeBillingGroup",
    "Resource": "arn:aws:iot:*:billinggroup/*",
    "Condition": {
        "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
    }
}
]
```

You can attach this policy to the IAM users in your account. If a user named `richard-roe` attempts to view an AWS IoT billing group, the billing group must be tagged `Owner=richard-roe` or `owner=richard-roe`. Otherwise, he is denied access. The condition tag key `Owner` matches both `Owner` and `owner` because condition key names are not case-sensitive. For more information, see [IAM JSON Policy Elements: Condition](#) in the [IAM User Guide](#).

## Troubleshooting AWS IoT Identity and Access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS IoT and IAM.

### Topics

- [I am not authorized to perform an action in AWS IoT \(p. 195\)](#)
- [I am not authorized to perform iam:PassRole \(p. 195\)](#)
- [I want to view my access keys \(p. 196\)](#)
- [I'm an administrator and want to allow others to access AWS IoT \(p. 196\)](#)
- [I want to allow people outside of my AWS account to access my AWS IoT resources \(p. 196\)](#)

## I am not authorized to perform an action in AWS IoT

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a thing but does not have `iot:DescribeThing` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
  iot:DescribeThing
      on resource: MyIoTThing
```

In this case, Mateo asks his administrator to update his policies to allow him to access the `MyIoTThing` resource using the `iot:DescribeThing` action.

## I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to AWS IoT.

Some AWS services allow you to pass an existing role to that service, instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in AWS IoT. However, the action requires the service to have permissions granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

## I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJAlrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

### Important

Do not provide your access keys to a third party, even to help [find your canonical user ID](#). By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see [Managing Access Keys](#) in the *IAM User Guide*.

## I'm an administrator and want to allow others to access AWS IoT

To allow others to access AWS IoT, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in AWS IoT.

To get started right away, see [Creating Your First IAM Delegated User and Group](#) in the *IAM User Guide*.

## I want to allow people outside of my AWS account to access my AWS IoT resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether AWS IoT supports these features, see [How AWS IoT Works with IAM](#) (p. 178).
- To learn how to provide access to your resources across AWS accounts that you own, see [Providing Access to an IAM User in Another AWS Account That You Own](#) in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see [Providing Access to AWS Accounts Owned by Third Parties](#) in the *IAM User Guide*.
- To learn how to provide access through identity federation, see [Providing Access to Externally Authenticated Users \(Identity Federation\)](#) in the *IAM User Guide*.

- To learn the difference between using roles and resource-based policies for cross-account access, see [How IAM Roles Differ from Resource-based Policies](#) in the *IAM User Guide*.

## Logging and Monitoring with AWS IoT Core

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 metrics:

- PublishIn.Success
- PublishOut.Success
- Subscribe.Success
- Ping.Success
- Connect.Success
- GetThingShadow.Accepted
- UpdateThingShadow.Accepted
- DeleteThingShadow.Accepted
- RulesExecuted

### Topics

- [Monitoring Tools \(p. 197\)](#)
- [Monitoring with Amazon CloudWatch \(p. 198\)](#)
- [Monitoring with CloudWatch Logs \(p. 209\)](#)
- [Logging AWS IoT API Calls with AWS CloudTrail \(p. 231\)](#)

## Monitoring Tools

AWS provides tools that you can use to monitor AWS IoT. You can configure some of these tools to do the monitoring for you. 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 Amazon EC2 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 \(p. 198\)](#).
- **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 service 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
  - Policies
  - Rules
  - Things
- CloudWatch home page shows:
  - Current alarms and status.
  - Graphs of alarms and resources.
  - Service health status.

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 sent automatically to CloudWatch in one minute

intervals. 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 \(p. 199\)](#)
- [How Do I Use AWS IoT Metrics? \(p. 207\)](#)
- [Creating CloudWatch Alarms to Monitor AWS IoT \(p. 207\)](#)

## AWS IoT Metrics and Dimensions

When you interact with AWS IoT, the service 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 (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, choose a metrics category, and then in the upper pane, scroll down to view the full list of metrics.

### To view metrics (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

The AWS/IoT namespace includes the following metrics. AWS IoT sends the following metrics to CloudWatch once per received request. For more information about CloudWatch metrics, see [Amazon CloudWatch Metrics](#).

### AWS IoT Metrics

| Metric                                | Description                                                                                                                                                                                  |
|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AddThingToDynamicThingGroupsFailed    | The number of failure events associated with adding a thing to a dynamic thing group. The DynamicThingGroupName dimension contains the name of the dynamic groups that failed to add things. |
| NumLogBatchesFailedToPublishThrottled | The singular batch of log events that has failed to publish due to throttling errors.                                                                                                        |
| NumLogEventsFailedToPublishThrottled  | The number of log events within the batch that have failed to publish due to throttling errors.                                                                                              |
| RulesExecuted                         | The number of AWS IoT rules executed.                                                                                                                                                        |

## Rule Metrics

| Metric               | Description                                                                                                                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ParseError           | The number of JSON parse errors that occurred in messages published on a topic on which a rule is listening. The RuleName dimension contains the name of the rule.                                                                    |
| RuleMessageThrottled | The number of messages throttled by the rules engine because of malicious behavior or because the number of messages exceeds the rules engine's throttle limit. The RuleName dimension contains the name of the rule to be triggered. |
| RuleNotFound         | The rule to be triggered could not be found. The RuleName dimension contains the name of the rule.                                                                                                                                    |
| 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.                                                                                              |

## Rule Action Metrics

| Metric  | Description                                                                                                                                                                                                                                                                          |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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. |
| 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.                                                                             |

## HTTP Action Specific Metrics

| Metric         | Description                                                                                                      |
|----------------|------------------------------------------------------------------------------------------------------------------|
| HttpCode_Other | Generated if the status code of the response from the downstream web service/application is not 2xx, 4xx or 5xx. |
| HttpCode_4XX   | Generated if the status code of the response from the downstream web service/application is between 400 and 499. |
| HttpCode_5XX   | Generated if the status code of the response from the downstream web service/application is between 500 and 599. |

| Metric             | Description                                                                                                                                                           |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HttpInvalidUrl     | Generated if an endpoint URL, after substitution templates are replaced, does not start with https://.                                                                |
| HttpRequestTimeout | Generated if the downstream web service/application does not return response within request timeout limit. For more information, see <a href="#">Service Quotas</a> . |
| HttpUnknownHost    | Generated if the URL is valid, but the service does not exist or is unreachable.                                                                                      |

## 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 <a href="#">AWS IoT Limits</a> . The Protocol dimension contains the protocol used to send the CONNECT message. |
| Connect.ClientIDThrottle | The number of connection requests throttled because the client exceeded the allowed connect request rate for a specific client ID. 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 account 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.                                                                         |
| PublishIn.ClientError    | The number of publish requests rejected by the message broker because the message did not meet the requirements defined in <a href="#">AWS IoT Limits</a> . The Protocol                                                     |

| Metric                 | Description                                                                                                                                                                                                                                               |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                        | 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 <a href="#">AWS IoT Limits</a> . 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 <a href="#">AWS IoT Limits</a> . 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 <code>Protocol</code> dimension contains the protocol used to send the <code>SUBSCRIBE</code> message.                                                           |
| Unsubscribe.ClientError | The number of unsubscribe requests that were rejected because the <code>UNSUBSCRIBE</code> message did not meet the requirements defined in <a href="#">AWS IoT Limits</a> . The <code>Protocol</code> dimension contains the protocol used to send the <code>UNSUBSCRIBE</code> message. |
| Unsubscribe.ServerError | The number of unsubscribe requests that were rejected because an internal error occurred. The <code>Protocol</code> dimension contains the protocol used to send the <code>UNSUBSCRIBE</code> message.                                                                                    |
| Unsubscribe.Success     | The number of unsubscribe requests that were successfully processed by the message broker. The <code>Protocol</code> dimension contains the protocol used to send the <code>UNSUBSCRIBE</code> message.                                                                                   |
| Unsubscribe.Throttle    | The number of unsubscribe requests that were rejected because the client exceeded the allowed unsubscribe request rate. The <code>Protocol</code> dimension contains the protocol used to send the <code>UNSUBSCRIBE</code> message.                                                      |

#### Note

The message broker metrics are displayed in the AWS IoT console under **Protocol Metrics**.

### Device Shadow Metrics

| Metric                     | Description                                                                                                                                                       |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DeleteThingShadow.Accepted | The number of <code>DeleteThingShadow</code> requests processed successfully. The <code>Protocol</code> dimension contains the protocol used to make the request. |
| GetThingShadow.Accepted    | The number of <code>GetThingShadow</code> requests processed successfully. The <code>Protocol</code> dimension contains the protocol used to make the request.    |
| UpdateThingShadow.Accepted | The number of <code>UpdateThingShadow</code> requests processed successfully. The <code>Protocol</code> dimension contains the protocol used to make the request. |

#### Note

The device shadow metrics are displayed in the AWS IoT console under **Protocol Metrics**.

### Jobs Metrics

| Metric                    | Description                                                                                                 |
|---------------------------|-------------------------------------------------------------------------------------------------------------|
| CanceledJobExecutionCount | The number of job executions whose status has changed to <code>CANCELED</code> within a time period that is |

| Metric                                        | Description                                                                                                                                                                                                                                                                                            |
|-----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                               | determined by CloudWatch. (For more information about CloudWatch metrics, see <a href="#">Amazon CloudWatch Metrics</a> .) The <code>JobId</code> dimension contains the ID of the job.                                                                                                                |
| <code>CanceledJobExecutionTotalCount</code>   | The total number of job executions whose status is <code>CANCELED</code> for the given job. The <code>JobId</code> dimension contains the ID of the job.                                                                                                                                               |
| <code>ClientError</code>                      | The number of client errors generated while executing the job. The <code>JobId</code> dimension contains the ID of the job.                                                                                                                                                                            |
| <code>FailedJobExecutionCount</code>          | The number of job executions whose status has changed to <code>FAILED</code> within a time period that is determined by CloudWatch. (For more information about CloudWatch metrics, see <a href="#">Amazon CloudWatch Metrics</a> .) The <code>JobId</code> dimension contains the ID of the job.      |
| <code>FailedJobExecutionTotalCount</code>     | The total number of job executions whose status is <code>FAILED</code> for the given job. The <code>JobId</code> dimension contains the ID of the job.                                                                                                                                                 |
| <code>InProgressJobExecutionCount</code>      | The number of job executions whose status has changed to <code>IN_PROGRESS</code> within a time period that is determined by CloudWatch. (For more information about CloudWatch metrics, see <a href="#">Amazon CloudWatch Metrics</a> .) The <code>JobId</code> dimension contains the ID of the job. |
| <code>InProgressJobExecutionTotalCount</code> | The total number of job executions whose status is <code>IN_PROGRESS</code> for the given job. The <code>JobId</code> dimension contains the ID of the job.                                                                                                                                            |
| <code>RejectedJobExecutionTotalCount</code>   | The total number of job executions whose status is <code>REJECTED</code> for the given job. The <code>JobId</code> dimension contains the ID of the job.                                                                                                                                               |
| <code>RemovedJobExecutionTotalCount</code>    | The total number of job executions whose status is <code>REMOVED</code> for the given job. The <code>JobId</code> dimension contains the ID of the job.                                                                                                                                                |
| <code>QueuedJobExecutionCount</code>          | The number of job executions whose status has changed to <code>QUEUED</code> within a time period that is determined by CloudWatch. (For more information about CloudWatch metrics, see <a href="#">Amazon CloudWatch Metrics</a> .) The <code>JobId</code> dimension contains the ID of the job.      |
| <code>QueuedJobExecutionTotalCount</code>     | The total number of job executions whose status is <code>QUEUED</code> for the given job. The <code>JobId</code> dimension contains the ID of the job.                                                                                                                                                 |

| Metric                          | Description                                                                                                                                                                                                                                                               |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RejectedJobExecutionCount       | The number of job executions whose status has changed to REJECTED within a time period that is determined by CloudWatch. (For more information about CloudWatch metrics, see <a href="#">Amazon CloudWatch Metrics</a> .) The JobId dimension contains the ID of the job. |
| RemovedJobExecutionCount        | The number of job executions whose status has changed to REMOVED within a time period that is determined by CloudWatch. (For more information about CloudWatch metrics, see <a href="#">Amazon CloudWatch Metrics</a> .) The JobId dimension contains the ID of the job.  |
| ServerError                     | The number of server errors generated while executing the job. The JobId dimension contains the ID of the job.                                                                                                                                                            |
| SucceededJobExecutionCount      | The number of job executions whose status has changed to SUCCESS within a time period that is determined by CloudWatch. (For more information about CloudWatch metrics, see <a href="#">Amazon CloudWatch Metrics</a> .) The JobId dimension contains the ID of the job.  |
| SucceededJobExecutionTotalCount | The total number of job executions whose status is SUCCESS for the given job. The JobId dimension contains the ID of the job.                                                                                                                                             |

### Device Defender Audit Metrics

| Metric                | Description                                                                                                                                                                             |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NonCompliantResources | The number of resources that were found to be noncompliant with a check. The system reports the number of resources that were out of compliance for each check of each audit performed. |
| ResourcesEvaluated    | The number of resources that were evaluated for compliance. The system reports the number of resources that were evaluated for each check of each audit performed.                      |

### Device Defender Detect Metrics

| Metric     | Description                                                                                                                                                                                                                                                                                    |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Violations | The number of new violations of security profile behaviors that have been found since the last time an evaluation was performed. The system reports the number of new violations for the account, for a specific security profile, and for a specific behavior of a specific security profile. |

| Metric                | Description                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ViolationsCleared     | The number of violations of security profile behaviors that have been resolved since the last time an evaluation was performed. The system reports the number of resolved violations for the account, for a specific security profile, and for a specific behavior of a specific security profile.                                                                                                                                  |
| ViolationsInvalidated | The number of violations of security profile behaviors for which information is no longer available since the last time an evaluation was performed (because the reporting device stopped reporting, or is no longer being monitored for some reason). The system reports the number of invalidated violations for the entire account, for a specific security profile, and for a specific behavior of a specific security profile. |

## Device Provisioning Metrics

| Metric                         | Description                                                                         |
|--------------------------------|-------------------------------------------------------------------------------------|
| CreateKeysAndCertificateFailed | The number of failures that occurred calling the CreateKeysAndCertificate MQTT API. |
| GetDeviceCredentialsSucceeded  | The number of successful GetDeviceCredentials calls.                                |
| GetDeviceCredentialsFailed     | The number of GetDeviceCredentials calls that failed.                               |
| DeviceProvisioningFailed       | The number of devices that failed to be provisioned                                 |
| DeviceProvisioningSucceeded    | The number of devices provisioned successfully.                                     |
| RegisterThingFailed            | The number of failures that occurred when calling the MQTT RegisterThing API.       |

## Dimensions for Metrics

Metrics use the namespace and provide metrics for the following dimensions:

| Dimension  | Description                                                                                |
|------------|--------------------------------------------------------------------------------------------|
| ActionType | The <a href="#">action type</a> specified by the rule that triggered 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.                                             |
| JobId      | The ID of the job whose progress or message connection success/failure is being monitored. |

| Dimension           | Description                                                                                                                                                                                 |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CheckName           | The name of the Device Defender audit check whose results are being monitored.                                                                                                              |
| ScheduledAuditName  | The name of the Device Defender scheduled audit whose check results are being monitored. This has the value OnDemand if the results reported are for an audit that was performed on demand. |
| SecurityProfileName | The name of the Device Defender Detect security profile whose behaviors are being monitored.                                                                                                |
| BehaviorName        | The name of the Device Defender Detect security profile behavior that is being monitored.                                                                                                   |
| TemplateName        | The name of the provisioning template.                                                                                                                                                      |

## 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? \(p. 207\)](#)
- [How can I be notified if my things are not publishing data each day? \(p. 208\)](#)
- [How can I be notified if my thing's shadow updates are being rejected each day? \(p. 208\)](#)

## 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 trigger actions for sustained state changes only. CloudWatch alarms do not trigger 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 [Getting Started with Amazon SNS](#).
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
```

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.
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
```

## Monitoring with CloudWatch Logs

AWS IoT sends progress events about each message as it passes from your devices through the message broker and 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](#). For information about supported AWS IoT CloudWatch Logs, see [CloudWatch Log Entry Format \(p. 214\)](#).

To enable AWS IoT logging, you must create an IAM role, register the role with AWS IoT, and then configure AWS IoT logging. In the CloudWatch console, CloudWatch logs appear in a log group named **AWSIoTLogs**.

### 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 role**.
2. Under **Choose the service that will use this role**, choose **AWS Service**.
3. Under **Select your use case**, choose **IoT**, and then choose **Next: Permissions**.
4. On the page that displays the policies that are automatically attached to the service role, choose **Next: Tags**.
5. Choose **Next: Review**.
6. Enter 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 use the AWS IoT console, the [set-v2-logging-options](#) CLI command, or the [SetV2LoggingOptions](#) API to enable logging. The principal used to make the API call must have [Pass Role Permissions](#) (p. 262) for your logging role. The logging role is passed to [set-v2-logging-options](#) or [SetV2LoggingOptions](#) as the `roleARN` parameter.

You can configure logging to be global or fine-grained. 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. You can use the AWS IoT console, the CLI, or the API to enable global logging. You must use the CLI or API to enable fine-grained logging.

### Global Logging

Use the `set-v2-logging-options` CLI command to set the logging options for your account. `set-v2-logging-options` takes three arguments:

`--role-arn`

Your logging role ARN. The logging role grants AWS IoT permission to write to your logs in CloudWatch Logs.

`--default-log-level`

The log level to use. Valid values are: ERROR, WARN, INFO, DEBUG, or DISABLED

`--disable-all-logs | --no-disable-all-logs`

When set to true (`--disable-all-logs`) disables all logs. The default (parameter not used) is false.

For example:

```
aws iot set-v2-logging-options \
--role-arn arn:aws:iam::<your-aws-account-num>:role/<IoTLoggingRole> \
--default-log-level <INFO>
```

You can use the `get-v2-logging-options` CLI command to get the current logging options.

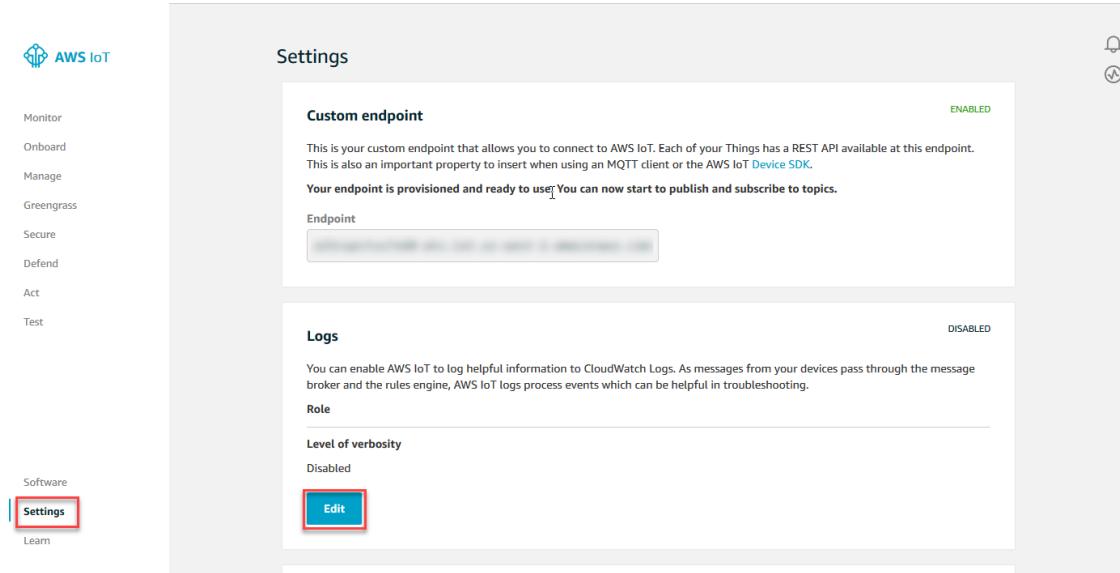
#### Note

AWS IoT continues to support older commands (`set-logging-options` and `get-logging-options`) to set and get global logging on your account. Be aware that when these commands

are used, the resulting logs contain plain-text, rather than JSON payloads and logging latency is generally higher. No further improvements will be made to the implementation of these older commands. We recommend that you use the "v2" versions to configure your logging options and, when possible, change legacy applications that use the older versions.

### To use the AWS IoT console to configure global logging

1. Sign in to the AWS IoT console. For more information, see [Sign in to the AWS IoT Console \(p. 5\)](#).
2. In the left navigation pane, choose **Settings**.



3. In the **Logs** section of the **Settings** page, choose **Edit**. The **Logs** section displays your settings for role and level of verbosity.
4. On the **Configure role setting** page, choose the level of verbosity that describes the level of detail that you want to appear in the CloudWatch logs.

The screenshot shows the 'Configure role setting' page. At the top, it says 'Configure role setting'. Below that is a 'Level of verbosity' section with a dropdown menu set to 'Disable logging', which is highlighted with a red box. Underneath is a 'Set role' section with a note: 'You can select a role to log specific account-level information to CloudWatch Logs.' It shows 'No role selected' and has 'Create Role' and 'Select' buttons, both highlighted with red boxes. At the bottom right is a large 'Update' button highlighted with a red box.

5. Choose **Select** to specify a role that you created previously, or **Create Role** to create a role to use for logging.
6. Choose **Update** to save your changes.

Review your CloudWatch logs to see if you are satisfied with the level of collected information. If not, you can always change the logging level later.

## 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 thing group. For example, you might have a thing group named "Phones" that contains things that represent different kinds of phones. You might then create another thing group named "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, there are two different logging levels assigned 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 overrides 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 \(p. 209\)](#).

`--default-log-level`

The logging level used if not specified. Valid values are: `DEBUG`, `INFO`, `ERROR`, `WARN`, and `DISABLED`.

`--disable-all-logs | --no-disable-all-logs`

When set to true (`--disable-all-logs`), disables all logs. The default (parameter not used) is false.

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 object that contains the resource type (field `targetType`) and name (field `targetName`) of the entity for which you are configuring logging. AWS IoT currently supports `THING_GROUP` for the 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 \(p. 214\)](#)
- [Device Shadow service \(p. 218\)](#)
- [Rules engine \(p. 220\)](#)
- [Jobs \(p. 225\)](#)
- [Device provisioning logs \(p. 229\)](#)
- [Dynamic Thing Groups Logs \(p. 230\)](#)

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" \(p. 210\)](#).

`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 Information \(1\)](#)

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 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.

**sourcelp**

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 Information \(2\)](#)

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 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

When the AWS IoT message broker receives an MQTT message, it generates a **Publish-In** log.

[More Information \(3\)](#)

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 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.

**sourcelp**

The IP address where the request originated.

**sourcePort**

The port where the request originated.

### Publish-Out Log

When the message broker publishes an MQTT message, it generates a Publish-Out log.

[More Information \(4\)](#)

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 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.

**sourcelp**

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 Information \(5\)](#)

For example:

```
{  
    "timestamp": "2017-08-10 15:37:23.476",  
    "logLevel": "INFO",  
    "traceId": "20b23f3f-d7f1-fae-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 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.

## Device Shadow Logs

The AWS IoT Device Shadow service generates logs for the following events:

[GetThingShadow Logs](#)

The Device Shadow service generates a `GetThingShadow` log when a get request for a shadow is received.

[More Information \(6\)](#)

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 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 shadow.

`topicName`

The name of the topic on which the request was published.

### UpdateThingShadow Logs

The Device Shadow service generates a `UpdateThingShadow` log when a request to update a device's shadow is received.

#### More Information (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 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 shadow to update.

#### topicName

The name of the topic on which the request was published.

#### DeleteThingShadow Logs

The Device Shadow service generates a `DeleteThingShadow` log when a request to delete a device's shadow is received.

##### More Information (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 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 shadow to update.

##### topicName

The name of the topic on which the request was published.

## Rules Engine Logs

The AWS IoT rules engine 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 Information (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 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 Information \(10\)](#)

A FunctionExecution log looks 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 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

When the AWS IoT rules engine starts to trigger a rule's action, it generates a StartingExecution log.

#### More Information (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 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 action triggered.

### principalId

The ID of the principal making the request.

## Rule Execution Logs

When the AWS IoT rules engine triggers a rule's action, it generates a `RuleExecution` log.

[More Information \(12\)](#)

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 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 action triggered.

### resources

A collection of resources used by the rule's actions.

### principalId

The ID of the principal making the request.

## Rule Not Found Logs

When the AWS IoT rules engine cannot find a rule with a given name, it generates a `RuleNotFound` error log.

## More Information (13)

For example:

```
{  
    "timestamp": "2017-10-04 19:25:46.070",  
    "logLevel": "ERROR",  
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",  
    "accountId": "123456789012",  
    "status": "Failure",  
    "eventType": "RuleNotFound",  
    "clientId": "abf27092886e49a8a5c1922749736453",  
    "topicName": "$aws/rules/example_rule",  
    "ruleName": "example_rule",  
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",  
    "reason": "RuleNotFound",  
    "details": "Rule example_rule not found"  
}
```

In addition to the attributes common to CloudWatch Logs, RuleNotFound log entries contain the following attributes:

eventType

RuleNotFound for rule-not-found logs.

clientId

The ID of the client making the request.

topicName

The name of the topic that was published.

ruleName

The name of the rule that could not be found.

principalId

The ID of the principal making the request.

reason

The string "RuleNotFound".

details

A brief explanation of the error.

## Rule Message Throttled Logs

When a message is throttled, the AWS IoT rules engine generates a RuleMessageThrottled error log.

## More Information (14)

For example:

```
{  
    "timestamp": "2017-10-04 19:25:46.070",  
    "logLevel": "ERROR",  
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",  
    "ruleName": "example_rule",  
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",  
    "topicName": "$aws/rules/example_rule",  
    "status": "Failure",  
    "eventType": "RuleMessageThrottled",  
    "clientId": "abf27092886e49a8a5c1922749736453",  
    "details": "Message throttled due to rate limit."}  
}
```

```
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleMessageThrottled",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "$aws/rules/example_rule",
    "ruleName": "example_rule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "reason": "RuleExecutionThrottled",
    "details": "Message for Rule example_rule throttled"
}
```

In addition to the attributes common to CloudWatch Logs, `RuleMessageThrottled` log entries contain the following attributes:

`eventType`

The value is `RuleMessageThrottled` for rule message throttled logs.

`clientId`

The ID of the client making the request.

`topicName`

The name of the topic that was published.

`ruleName`

The name of the rule to be triggered.

`principalId`

The ID of the principal making the request.

`reason`

The string `"RuleMessageThrottled"`.

`details`

A brief explanation of the error.

## Job Logs

The AWS IoT Job service generates logs for the following events. Logs are generated when an MQTT or HTTP request is received from the device.

### Get Pending Job Execution Logs

The AWS IoT Jobs service generates a `GetJobExecution` log when the service receives a job execution request.

#### More Information (16)

For example:

```
{
    "timestamp": "2018-06-13 17:45:17.197",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "GetPendingJobExecution",
    "protocol": "MQTT",
```

```
        "clientId": "299966ad-54de-40b4-99d3-4fc8b52da0c5",
        "topicName": "$aws/things/299966ad-54de-40b4-99d3-4fc8b52da0c5/jobs/get",
        "clientToken": "24b9a741-15a7-44fc-bd3c-1ff2e34e5e82",
        "details": "The request status is SUCCESS."
    }
```

In addition to the attributes common to CloudWatch Logs, `GetPendingJobExecution` log entries contain the following attributes:

**eventType**

`GetPendingJobExecution` for get pending 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.

**clientToken**

A unique, case sensitive identifier to ensure the idempotency of the request. For more information, see [How to Ensure Idempotency](#).

**details**

Other information from the Jobs service.

## Describe Job Execution Logs

The AWS IoT Jobs service generates a `DescribeJobExecution` log when the service receives a request to describe a job execution.

### More Information (17)

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 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**

A unique, case sensitive identifier to ensure the idempotency of the request. For more information, see [How to Ensure Idempotency](#).

**details**

Other information from the Jobs service.

## Update Job Execution Logs

The AWS IoT Jobs service generates an `UpdateJobExecution` log when the service receives a request to update a job execution.

More Information (18)

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 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

A unique, case sensitive identifier to ensure the idempotency of the request. For more information, see [How to Ensure Idempotency](#).

versionNumber

The version of the job execution.

details

Other information from the Jobs service.

### Start Next Pending Job Execution Logs

When it receives a request to start the next pending job execution, the AWS IoT Jobs service generates a `StartNextPendingJobExecution` log.

[More Information \(19\)](#)

For example:

```
{  
    "timestamp": "2018-06-13 17:49:51.036",  
    "logLevel": "DEBUG",  
    "accountId": "123456789012",  
    "status": "Success",  
    "eventType": "StartNextPendingJobExecution",  
    "protocol": "MQTT",  
    "clientId": "95c47808-b1ca-4794-bc68-a588d6d9216c",  
    "topicName": "$aws/things/95c47808-b1ca-4794-bc68-a588d6d9216c/jobs/start-next",  
    "clientToken": "bd7447c4-3a05-49f4-8517-dd89b2c68d94",  
    "details": "The request status is SUCCESS."  
}
```

In addition to the attributes common to CloudWatch Logs, `StartNextPendingJobExecution` log entries contain the following attributes:

eventType

`StartNextPendingJobExecution` for start-next-pending-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 topic used to make the request.

clientToken

A unique, case sensitive identifier to ensure the idempotency of the request. For more information, see [How to Ensure Idempotency](#).

details

Other information from the Jobs service.

## Report Final Job Execution Count Logs

The AWS IoT Jobs service generates a `ReportFinalJobExecutionCount` log when a job is completed.

[More Information \(20\)](#)

For example:

```
{  
    "timestamp": "2017-08-10 19:44:16.776",  
    "logLevel": "INFO",  
    "accountId": "123456789012",  
    "status": "Success",  
    "eventType": "ReportFinalJobExecutionCount",  
    "jobId": "002",  
    "details": "Job 002 completed. QUEUED job execution count: 0 IN_PROGRESS job  
execution count: 0 FAILED job execution count: 0 SUCCEEDED 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 CloudWatch Logs, `ReportFinalJobExecutionCount` log entries contain the following attributes:

`eventType`

`ReportFinalJobExecutionCount` for report-final-job-execution-count logs.

`jobId`

The job ID for the job execution.

`details`

Other information from the Jobs service.

## Device Provisioning Logs

The AWS IoT Device Provisioning service generates logs for the following events.

[GetDeviceCredentials Logs](#)

The AWS IoT Device Provisioning service generates a log when a client calls `GetDeviceCredential`.

[more info \(16\)](#)

For example:

```
{  
    "timestamp" : "2019-02-20 20:31:22.932",  
    "logLevel" : "INFO",  
    "traceId" : "8d9c016f-6cc7-441e-8909-7ee3d5563405",  
    "accountId" : "123456789101",  
    "status" : "Success",  
    "eventType" : "GetDeviceCredentials",  
    "deviceCertificateId" :  
        "e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855",  
    "details" : "Additional details about this log."  
}
```

## ProvisionDevice Logs

The AWS IoT Device Provisioning service generates a log when a client calls `ProvisionDevice`.  
[more info \(16\)](#)

For example:

```
{  
    "timestamp" : "2019-02-20 20:31:22.932",  
    "logLevel" : "INFO",  
    "traceId" : "8d9c016f-6cc7-441e-8909-7ee3d5563405",  
    "accountId" : "123456789101",  
    "status" : "Success",  
    "eventType" : "ProvisionDevice",  
    "provisioningTemplateName" : "myTemplate",  
    "deviceCertificateId" :  
        "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",  
    "details" : "Additional details about this log."  
}
```

## Dynamic Thing Group Logs

AWS IoT Dynamic Thing Groups generate logs for the following event.

`AddThingToDynamicThingGroupsFailed` event

AWS IoT was not able to add a thing to the specified dynamic groups.

A thing met the criteria to be in the dynamic thing group; however, it could not be added to the dynamic group or it was removed from the dynamic group. This can happen because:

- The thing already belongs to the maximum number of groups.
- The **--override-dynamic-groups** option was used to add the thing to a static thing group. It was removed from a dynamic thing group to make that possible.

For more information, see [Dynamic Thing Group Limitations and Conflicts](#).

[More info \(17\)](#)

This example shows the log entry of an `AddThingToDynamicThingGroupsFailed` error. In this example, `TestThing` met the criteria to be in the dynamic thing groups listed in `dynamicThingGroupNames`, but could not be added to those dynamic groups, as described in `reason`.

```
{  
    "timestamp": "2020-03-16 22:24:43.804",  
    "logLevel": "ERROR",  
    "traceId": "70b1f2f5-d95e-f897-9dcc-31e68c3e1a30",  
    "accountId": "571032923833",  
    "status": "Failure",  
    "eventType": "AddThingToDynamicThingGroupsFailed",  
    "thingName": "TestThing",  
    "dynamicThingGroupNames": [  
        "DynamicThingGroup11",  
        "DynamicThingGroup12",  
        "DynamicThingGroup13",  
        "DynamicThingGroup14"  
    ],  
    "reason": "The thing failed to be added to the given dynamic thing group(s) because  
the thing already belongs to the maximum allowed number of groups."  
}
```

}

thingName

The name of the thing that could not be added to a dynamic thing group.  
dynamicThingGroupNames

An array of the dynamic thing groups to which the thing could not be added.  
reason

The reason why the thing could not be added to the dynamic thing groups.

## 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, enter **AWSIoTLogsV2**, and then 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 enter 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](#).

## Logging AWS IoT API Calls with AWS CloudTrail

AWS IoT is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS IoT. CloudTrail captures all API calls for AWS IoT as events, including calls from the AWS IoT console and from code calls to the AWS IoT APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS IoT. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to AWS IoT, the IP address from which the request was made, who made the request, when it was made, and other details.

To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

## AWS IoT Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS IoT, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**.

You can view, search, and download recent events in your AWS account. For more information, see [Viewing Events with CloudTrail Event History](#).

For an ongoing record of events in your AWS account, including events for AWS IoT, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all AWS Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. You can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see:

- [Overview for Creating a Trail](#)
- [CloudTrail Supported Services and Integrations](#)
- [Configuring Amazon SNS Notifications for CloudTrail](#)
- [Receiving CloudTrail Log Files from Multiple Regions](#) and [Receiving CloudTrail Log Files from Multiple Accounts](#)

**Note**

AWS IoT data plane actions (device side) are not logged by CloudTrail. Use CloudWatch to monitor these actions.

AWS IoT control plane actions are logged by CloudTrail. For example, calls to the **CreateThing**, **ListThings**, and **ListTopicRules** sections generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or IAM user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the [CloudTrail userIdentity Element](#). AWS IoT actions are documented in the [AWS IoT API Reference](#).

## Understanding AWS IoT Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files 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::123456789012: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": ""
}
```

## Compliance Validation for AWS IoT Core

Third-party auditors assess the security and compliance of AWS IoT Core as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see [AWS Services in Scope by Compliance Program](#). For general information, see [AWS Compliance Programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

Your compliance responsibility when using AWS IoT is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- [Security and Compliance Quick Start Guides](#) – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- [Architecting for HIPAA Security and Compliance Whitepaper](#) – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- [AWS Compliance Resources](#) – This collection of workbooks and guides might apply to your industry and location.

- [AWS Config](#) – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

## Resilience in AWS IoT Core

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

AWS IoT Core stores information about your devices in the device registry. It also stores CA certificates, device certificates, and device shadow data. This data is not automatically replicated in the event of hardware or network failures. AWS IoT Core publishes MQTT events when the device registry is updated. You can use these messages to back up your registry data and save it somewhere, like a DynamoDB table. You are responsible for saving certificates that AWS IoT Core creates for you or those you create yourself. Device shadow stores state data about your devices and can be resent when a device comes back online.

## Infrastructure Security in AWS IoT

As a collection of managed services, AWS IoT is protected by the AWS global network security procedures that are described in the [Amazon Web Services: Overview of Security Processes](#) whitepaper.

You use AWS published API calls to access AWS IoT through the network. Clients must support Transport Layer Security (TLS) 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems, such as Java 7 and later, support these modes. For more information, see [Transport Security in AWS IoT \(p. 173\)](#).

Requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the [AWS Security Token Service](#) (AWS STS) to generate temporary security credentials to sign requests.

## Vulnerability Analysis and Management in AWS IoT Core

IoT fleets can consist of large numbers of devices that have diverse capabilities, are long-lived, and are geographically distributed. These characteristics make fleet setup complex and error-prone. And because devices are often constrained in computational power, memory, and storage capabilities, this limits the use of encryption and other forms of security on the devices themselves. Also, devices often use software with known vulnerabilities. These factors make IoT fleets an attractive target for hackers and make it difficult to secure your device fleet on an ongoing basis.

AWS IoT Device Defender addresses these challenges by providing tools to identify security issues and deviations from best practices. You can use AWS IoT Device Defender to analyze, audit, and monitor connected devices to detect abnormal behavior, and mitigate security risks. AWS IoT Device Defender

can audit device fleets to ensure they adhere to security best practices and detect abnormal behavior on devices. This makes it possible to enforce consistent security policies across your AWS IoT device fleet and respond quickly when devices are compromised. For more information, see [AWS IoT Device Defender \(p. 534\)](#).

## Security Best Practices in AWS IoT Core

This section contains information about security best practices for AWS IoT Core. For more information, see [Ten security golden rules for IoT solutions](#).

### Protecting MQTT Connections in AWS IoT

[AWS IoT Core](#) is a managed cloud service that makes it possible for connected devices to interact with cloud applications and other devices easily and securely. AWS IoT Core supports HTTP, [WebSocket](#), and [MQTT](#), a lightweight communication protocol specifically designed to tolerate intermittent connections. If you are connecting to the [AWS IoT message broker](#) using MQTT, each of your connections must be associated with an identifier known as a client ID. MQTT client IDs uniquely identify MQTT connections. If a new connection is established using a client ID that is already claimed for another connection, the AWS IoT message broker drops the old connection to allow the new connection. Client IDs must be unique within each AWS account and each AWS Region. This means that you don't need to enforce global uniqueness of client IDs outside of your AWS account or across Regions within your AWS account.

The impact and severity of dropping MQTT connections on your device fleet depends on many factors. These include:

- Your use case (for example, the data your devices send to AWS IoT, how much data, and the frequency that the data is sent).
- Your MQTT client configuration (for example, auto reconnect settings, associated back-off timings, and use of [MQTT persistent sessions](#)).
- Device resource constraints.
- The root cause of the disconnections, its aggressiveness, and persistence.

To avoid client ID conflicts and their potential negative impacts, make sure that each device or mobile application has an AWS IoT or IAM policy that restricts which client IDs can be used for MQTT connections to the AWS IoT message broker. For example, you can use an IAM policy to prevent a device from unintentionally closing another device's connection by using a client ID that is already in use. For more information, see [Authorization \(p. 137\)](#).

All devices in your fleet must have credentials with privileges that authorize intended actions only, which include (but not limited to) AWS IoT MQTT actions such as publishing messages or subscribing to topics with specific scope and context. The specific permission policies can vary for your use cases. Identify the permission policies that best meet your business and security requirements.

To simplify creation and management of permission policies, you can use [AWS IoT Core Policy Variables \(p. 142\)](#) and [IAM policy variables](#). Policy variables can be placed in a policy and when the policy is evaluated, the variables are replaced by values that come from the device's request. Using policy variables, you can create a single policy for granting permissions to multiple devices. You can identify the relevant policy variables for your use case based on your AWS IoT account configuration, authentication mechanism, and network protocol used in connecting to AWS IoT message broker. However, to write the best permission policies, consider the specifics of your use case and your [threat model](#).

For example, if you registered your devices in the [AWS IoT registry](#), you can use [thing policy variables](#) in AWS IoT policies to 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](#). You can use the [AttachThingPrincipal](#) API to attach certificates and authenticated Amazon Cognito identities to a thing. `iot:Connection.Thing.ThingName` is a useful thing policy variable to enforce client ID restrictions. The following example AWS IoT policy requires a registered thing's name to be used as the client ID for MQTT connections to the AWS IoT message broker:

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

If you want to identify ongoing client ID conflicts, you can enable and use [CloudWatch Logs for AWS IoT](#). For every MQTT connection that the AWS IoT message broker disconnects due to client ID conflicts, a log record similar to the following is generated:

```
{
    "timestamp": "2019-04-28 22:05:30.105",
    "logLevel": "ERROR",
    "traceId": "02a04a93-0b3a-b608-a27c-1ae8ebdb032a",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "Disconnect",
    "protocol": "MQTT",
    "clientId": "clientId01",
    "principalId": "1670fcf6de55adc1930169142405c4a2493d9eb5487127cd0091ca0193a3d3f6",
    "sourceIp": "203.0.113.1",
    "sourcePort": 21335,
    "reason": "DUPLICATE_CLIENT_ID",
    "details": "A new connection was established with the same client ID"
}
```

You can use a [CloudWatch Logs filter](#) such as `{$.reason= "DUPLICATE_CLIENT_ID" }` to search for instances of client ID conflicts or to set up [CloudWatch metric filters](#) and corresponding CloudWatch alarms for continuous monitoring and reporting.

You can use [AWS IoT Device Defender](#) to identify overly permissive AWS IoT and IAM policies. AWS IoT Device Defender also provides an audit check that notifies you if multiple devices in your fleet are connecting to the AWS IoT message broker using the same client ID.

## See Also

- [AWS IoT Core](#)
- [AWS IoT's Security Features](#)
- [AWS IoT Policy Variables](#)
- [IAM Policy Variables](#)

- [Amazon Cognito Identity](#)
- [AWS IoT Registry](#)
- [AWS IoT Device Defender](#)
- [CloudWatch Logs for AWS IoT](#)

## Keep Your Device's Clock in Sync

It's important to have an accurate time on your device. X.509 certificates have an expiry date and time. The clock on your device is used to verify that a server certificate is still valid. If you're building commercial IoT devices, remember that your products may be stored for extended periods before being sold. Real-time clocks can drift during this time and batteries can get discharged, so setting time in the factory is not sufficient.

For most systems, this means that the device's software must include a network time protocol (NTP) client. The device should wait until it synchronizes with an NTP server before it tries to connect to AWS IoT Core. If this isn't possible, the system should provide a way for a user to set the device's time so that subsequent connections succeed.

After the device synchronizes with an NTP server, it can open a connection with AWS IoT Core. How much clock skew that is allowed depends on what you're trying to do with the connection.

## Validate the Server Certificate

The first thing a device does to interact with AWS IoT is to open a secure connection. When you connect your device to AWS IoT, ensure that you're talking to AWS IoT and not another server impersonating AWS IoT. Each of the AWS IoT servers is provisioned with a certificate issued for the `iot.amazonaws.com` domain. This certificate was issued to AWS IoT by a trusted certificate authority that verified our identity and ownership of the domain.

One of the first things AWS IoT Core does when a device connects is send the device a server certificate. Devices can verify that they were expecting to connect to `iot.amazonaws.com` and that the server on the end of that connection possesses a certificate from a trusted authority for that domain.

TLS certificates are in X.509 format and include a variety of information such as the organization's name, location, domain name, and a validity period. The validity period is specified as a pair of time values called `notBefore` and `notAfter`. Services like AWS IoT Core use limited validity periods (for example, one year) for their server certificates and begin serving new ones before the old ones expire.

## Use a Single Identity Per Device

Use a single identity per client. Devices generally use X.509 client certificates. Web and mobile applications use Amazon Cognito Identity. This enables you to apply fine-grained permissions to your devices.

For example, you have an application that consists of a mobile phone device that receives status updates from two different smart home objects – a light bulb and a thermostat. The light bulb sends the status of its battery level, and a thermostat sends messages that report the temperature.

AWS IoT authenticates devices individually and treats each connection individually. You can apply fine-grained access controls using authorization policies. You can define a policy for the thermostat that allows it to publish to a topic space. You can define a separate policy for the light bulb that allows it to

publish to a different topic space. Finally, you can define a policy for the mobile app that only allows it to connect and subscribe to the topics for the thermostat and the light bulb to receive messages from these devices.

Apply the principle of least privilege and scope down the permissions per device as much as possible. All devices or users should have an AWS IoT policy in AWS IoT that only allows it to connect with a known client ID, and to publish and subscribe to an identified and fixed set of topics.

## Use Just In Time Provisioning

Manually creating and provisioning each device can be time consuming. AWS IoT provides a way to define a template to provision devices when they first connect to AWS IoT. For more information, see [Just-in-Time Provisioning \(p. 502\)](#).

## AWS Training and Certification

Take the following course to learn about key concepts for AWS IoT security: [AWS IoT Security Primer](#).

# Connecting Devices

Devices connect to AWS IoT using a fully qualified domain name (FQDN) that is specific to your account. All connections to AWS IoT must be secured with TLS version 1.2 and AWS IoT requires devices to send the [Server Name Indication \(SNI\) extension](#). For more information, see [Transport Security in AWS IoT](#).

AWS IoT provides three types of endpoints, which are distinguished by the following three service types that they expose:

- Data – Used to send and receive data to and from the [Message Broker](#), [Device Shadow](#), and [Rules Engine](#) components of AWS IoT.
- Credential\_Provider – Used to exchange a device's built-in X.509 certificate for temporary credentials to connect directly with other AWS services. For more information about connecting to other AWS services, see [Authorizing Direct Calls to AWS Services](#).
- Jobs – Used to enable devices to interact with the AWS IoT Jobs service using the [Jobs Device HTTPS APIs](#).

Every AWS IoT customer has default endpoints for each service type. You use the [DescribeEndpoint API](#) to get your default endpoints. The following list contains the valid endpoint types:

- `iot:Data-ATS` – Endpoint for the Data service type.
- `iot:CredentialProvider` – Endpoint for the Credential\_Provider service type.
- `iot:Jobs` – Endpoint for the Jobs service type.

## Important

Every customer has another endpoint type: `iot:Data`. Use this endpoint to retrieve old data endpoints that use VeriSign certificates for backward compatibility. We strongly recommend that customers use the newer `iot:Data-ATS` endpoint type to avoid issues related to the widespread distrust of Symantec certificate authorities. For more information, see [Server Authentication](#).

Specify the endpoint type to get from the [DescribeEndpoint API](#) by using the `endpointType` parameter. The following AWS CLI example gets your default `iot:Data-ATS` endpoint.

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

This command returns an endpoint in the following format: `account-specific-prefix-ats.iot.region.amazonaws.com`

You can also use your own FQDN (for example, `example.com`) and the associated server certificate to connect devices to AWS IoT by using the configurable endpoints feature, which is currently in public beta. The following section explains how to use configurable endpoints for your custom domains and AWS-managed domains.

## Topics

- [Configurable Endpoints \(Beta\) \(p. 239\)](#)

## Configurable Endpoints (Beta)

This feature is currently in public beta and is available only in the US East (N. Virginia) Region.

You can create multiple endpoints for connecting devices to AWS IoT. You can also configure some details (such as domain name) by using domain configurations. You can use domain configurations to simplify tasks such as the following.

- Migrate devices to AWS IoT.
- Support heterogeneous device fleets by maintaining separate domain configurations for separate device types.
- Maintain brand identity (for example, through domain name) while migrating application infrastructure to AWS IoT.

In the beta release, you can configure a fully qualified domain name (FQDN) and the associated server certificate too. You can also associate a custom authorizer or an enhanced custom authorizer. For more information, see [Custom Authorizers \(p. 130\)](#) and [Enhanced Custom Authentication \(Beta\) \(p. 134\)](#).

**Note**

AWS IoT uses the server name indication (SNI) TLS extension to apply domain configurations. Devices must use this extension when connecting and pass a server name that is identical to the domain name that is specified in the domain configuration. To test this service, use the v2 version of each [AWS IoT Device SDK in GitHub](#).

**Topics**

- [Creating and Configuring AWS-Managed Domains \(p. 240\)](#)
- [Creating and Configuring Custom Domains \(p. 241\)](#)
- [Managing Domain Configurations \(p. 243\)](#)

## Creating and Configuring AWS-Managed Domains

This feature is currently in public beta and is available only in the US East (N. Virginia) Region.

You create a configurable endpoint on an AWS-managed domain by using the [CreateDomainConfiguration](#) API. A domain configuration for an AWS-managed domain consists of the following:

- `domainConfigurationName` – A user-defined name that identifies the domain configuration.

**Note**

Domain configuration names that start with `IoT:` are reserved for default endpoints and can't be used. Also, this value must be unique to your region.

- `defaultAuthorizerName` – The name of the custom authorizer to use on the endpoint.
- `allowAuthorizerOverride` – A Boolean value that specifies whether devices can override the default authorizer by specifying a different authorizer in the HTTP header of the request. This value is required if a value for `defaultAuthorizerName` is specified.
- `serviceType` – Possible values are `DATA`, `CREDENTIAL_PROVIDER`, and `JOB`. If you specify `DATA`, AWS IoT returns an endpoint with an endpoint type of `iot:Data-Beta`. This is a special endpoint type for the configurable endpoints beta release. The endpoint serves ATS-signed server certificates. You can't create a configurable `iot:Data` (VeriSign) endpoint.

The following AWS CLI command creates domain configuration for a `DATA` endpoint.

```
aws iot create-domain-configuration --domain-configuration-name "myDomainConfigurationName" --service-type "DATA"
```

# Creating and Configuring Custom Domains

This feature is currently in public beta and is available only in the US East (N. Virginia) Region.

Domain configurations let you specify a custom fully qualified domain name (FQDN) to connect to AWS IoT. Custom domains enable you to manage your own server certificates so that you can manage details, such as the root certificate authority (CA) used to sign the certificate, the signature algorithm, the certificate chain depth, and the lifecycle of the certificate.

The workflow to set up a domain configuration with a custom domain consists of the following three stages.

1. [Registering Server Certificates in AWS Certificate Manager \(p. 241\)](#)
2. [Creating a Domain Configuration \(p. 242\)](#)
3. [Creating DNS Records \(p. 243\)](#)

## Registering Server Certificates in AWS Certificate Manager

Before you create a domain configuration with a custom domain, you must register your server certificate chain in [AWS Certificate Manager \(ACM\)](#). You can use three types of server certificates.

- [ACM Generated Public Certificates \(p. 241\)](#)
- [External Certificates Signed by a Public CA \(p. 241\)](#)
- [External Certificates Signed by a Private CA \(p. 242\)](#)

### Note

AWS IoT considers a certificate to be signed by a public CA if it's included in [Mozilla's trusted ca-bundle](#).

### Using One Certificate for Multiple Domains

If you plan to use one certificate to cover multiple subdomains, use a wildcard domain in the common name (CN) or Subject Alternative Names (SAN) field. For example, use `*.iot.example.com` to cover `dev.iot.example.com`, `qa.iot.example.com`, and `prod.iot.example.com`. Each FQDN requires its own domain configuration, but more than one domain configuration can use the same wildcard value. Either the CN or the SAN must cover the FQDN that you want to use as a custom domain. This coverage can be an exact match or a wildcard match.

The following sections describe how to get each type of certificate. Every certificate resource requires an ARN registered with ACM that you use when you create your domain configuration.

### ACM-Generated Public Certificates

You can generate a public certificate for your custom domain by using the [RequestCertificate API](#). When you generate a certificate in this way, ACM validates your ownership of the custom domain. For more information, see [Request a Public Certificate](#) in the [AWS Certificate Manager User Guide](#).

### External Certificates Signed by a Public CA

If you already have a server certificate that is signed by a public CA (a CA that is included in Mozilla's trusted ca-bundle), you can import the certificate chain directly into ACM by using the [ImportCertificate](#)

API. To learn more about this task and the prerequisites and certificate format requirements, see [Importing Certificates](#).

## External Certificates Signed by a Private CA

If you already have a server certificate that is signed by a private CA or self-signed, you can use the certificate to create your domain configuration, but you also must create an extra public certificate in ACM to validate ownership of your domain. To do this, register your server certificate chain in ACM using the [ImportCertificate API](#). To learn more about this task and the prerequisites and certificate format requirements, see [Importing Certificates](#).

After you import your certificate to ACM, generate a public certificate for your custom domain by using the [RequestCertificate API](#). When you generate a certificate in this way, ACM validates your ownership of the custom domain. For more information, see [Request a Public Certificate](#). When you create your domain configuration, use this public certificate as your validation certificate.

## Creating a Domain Configuration

You create a configurable endpoint on an custom domain by using the [CreateDomainConfiguration API](#). A domain configuration for a custom domain consists of the following:

- `domainConfigurationName` – A user-defined name that identifies the domain configuration.

**Note**

Domain configuration names starting with `IoT`: are reserved for default endpoints and can't be used. Also, this value must be unique to your region.

- `domainName` – The FQDN that your devices use to connect to AWS IoT.

**Note**

AWS IoT leverages the server name indication (SNI) TLS extension to apply domain configurations. Devices must use this extension when connecting and pass a server name that is identical to the domain name that is specified in the domain configuration.

- `serverCertificateArns` – The ARN of the server certificate chain that you registered with ACM. The beta release supports only one server certificate.
- `validationCertificateArn` – The ARN of the public certificate that you generated in ACM to validate ownership of your custom domain. This argument isn't required if you use a publicly signed or ACM-generated server certificate.
- `defaultAuthorizerName` – The name of the custom authorizer to use on the endpoint.
- `allowAuthorizerOverride` – A Boolean value that specifies whether devices can override the default authorizer by specifying a different authorizer in the HTTP header of the request. This value is required if a value for `defaultAuthorizerName` is specified.
- `serviceType` – Possible values are `DATA`, `CREDENTIAL_PROVIDER`, and `JOB`. If you specify `DATA`, AWS IoT returns an endpoint with an endpoint type of `iot:Data-Beta`. This is a special endpoint type for the configurable endpoints beta release. You can't create a configurable `iot:Data` (VeriSign) endpoint.

The following AWS CLI command creates a domain configuration for `iot.example.com`.

```
aws iot create-domain-configuration --domain-configuration-name "myDomainConfigurationName"
--service-type "DATA"
--domain-name "iot.example.com" --server-certificate-arns serverCertARN --validation-
certificate-arn validationCertArn
```

**Note**

After you create your domain configuration, it might take up to 15 minutes until AWS IoT serves your custom server certificates.

## Creating DNS Records

After you register your server certificate chain and create your domain configuration, create a DNS record so that your custom domain points to an AWS IoT domain. This record must point to an AWS IoT endpoint of type `iot:Data-Beta`. This is a special endpoint type for the configurable endpoints beta release. You can get your beta endpoint by using the [DescribeEndpoint](#) API.

The following AWS CLI command shows how to get your beta endpoint.

```
aws iot describe-endpoint --endpoint-type iot:Data-Beta
```

After you get your `iot:Data-Beta` endpoint, create a CNAME record from your custom domain to this AWS IoT endpoint. If you create multiple custom domains in the same account, alias them to this same `iot:Data-Beta` endpoint.

## Managing Domain Configurations

This feature is currently in public beta and is available only in the US East (N. Virginia) Region.

You can manage the lifecycles of existing configurations by using the following APIs.

- [ListDomainConfigurations](#)
- [DescribeDomainConfiguration](#)
- [UpdateDomainConfiguration](#)
- [DeleteDomainConfiguration](#)

## Viewing Domain Configurations

Use the [ListDomainConfigurations](#) API to return a paginated list of all domain configurations in your account. You can see the details of a particular domain configuration using the [DescribeDomainConfiguration](#) API. This API takes a single `domainConfigurationName` parameter and returns the details of the specified configuration.

## Updating Domain Configurations

To update the status or the custom authorizer of your domain configuration, use the [UpdateDomainConfiguration](#) API. You can set the status to `ENABLED` or `DISABLED`. If you disable the domain configuration, devices connected to that domain receive an authentication error.

**Note**

Currently you can't update the server certificate in your domain configuration. To change the certificate of a domain configuration, you must delete and recreate it.

## Deleting Domain Configurations

Before you delete a domain configuration, use the [UpdateDomainConfiguration](#) API to set the status to `DISABLED`. This helps you avoid accidentally deleting the endpoint. After you disable the domain configuration, delete it by using the [DeleteDomainConfiguration](#) API.

After you delete a domain configuration, AWS IoT no longer serves the server certificate associated with that custom domain.

# Message Broker for AWS IoT

The AWS IoT message broker connects AWS IoT clients by sending messages from publishing clients to subscribing clients. Clients send data by publishing a message on a topic and receive messages by subscribing to a topic. When the message broker receives a message from a publishing client, it forwards the message to all clients that have subscribed to that topic.

Processing messages through the AWS IoT message broker lets you define [rules](#) that can initiate more actions based on the content of the messages. If you do not require AWS IoT features such as [rules](#) or [jobs](#), see [AWS Messaging](#) for information about other AWS messaging services that might better fit your requirements.

## Topics

- [Topics \(p. 244\)](#)
- [Protocols \(p. 253\)](#)

## Topics

Topics identify AWS IoT messages. AWS IoT clients identify the messages they publish by giving the messages topic names. Clients identify the messages to which they want to subscribe (receive) by registering a topic filter with AWS IoT Core. The AWS IoT message broker uses topic names and topic filters to route messages from publishing clients to subscribing clients.

While AWS IoT supports some [reserved system topics](#), most MQTT topics are created and managed by you, the system designer. AWS IoT uses topics to identify messages received from publishing clients and select messages to send to subscribing clients, as described in the following sections. Before you create a topic namespace for your system, review the characteristics of MQTT topics to create the hierarchy of topic names that works best for your IoT system.

## Topic Names

Topic names and topic filters are UTF-8 encoded strings. They can represent a hierarchy of information by using the forward slash (/) character to separate the levels of the hierarchy. For example, this topic name could refer to a temperature sensor in room 1:

- `sensor/temperature/room1`

In this example, there might also be other types of sensors in other rooms with topic names such as:

- `sensor/temperature/room2`
- `sensor/humidity/room1`
- `sensor/humidity/room2`

### Note

As you consider topic names for the messages in your system, keep in mind:

- Topic names and topic filters are case sensitive.
- Topic names must not contain personally identifiable information.
- Topic names that begin with a \$ are [reserved topics](#) to be used only by AWS IoT Core.
- AWS IoT Core cannot send or receive messages between AWS accounts or Regions.

The topic namespace is limited to an AWS account and Region. For example, the `sensor/temp/room1` topic used by an AWS account in one Region is distinct from the `sensor/temp/room1` topic used by the same AWS account in another Region or used by any other AWS account in any Region.

## Topic Filters

Subscribing clients register topic filters with the AWS IoT message broker to specify the message topics that the message broker should send to them. A topic filter can be a single topic name to subscribe to a single topic name or it can include wildcard characters to subscribe to multiple topic names at once.

Publishing clients cannot use wildcard characters in the topic names they publish.

The following table lists the wildcard characters that can be used in a topic filter.

### Topic Wildcards

| Wildcard Character | Matches                                                    | Notes                                                                                                                                                                                                    |
|--------------------|------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #                  | All strings at and below its level in the topic hierarchy. | Must be the last character in the topic filter.<br><br>Must be the only character in its level of the topic hierarchy.<br><br>Can be used in a topic filter that also contains the + wildcard character. |
| +                  | Any string in the level that contains the character.       | Must be the only character in its level of the topic hierarchy.<br><br>Can be used in multiple levels of a topic filter.                                                                                 |

Using wildcards with the previous sensor topic name examples:

- A subscription to `sensor/#` receives messages published to `sensor/`, `sensor/temperature`, `sensor/temperature/room1`, but not messages published to `Sensor`.
- A subscription to `sensor/+/room1` receives messages published to `sensor/temperature/room1` and `sensor/humidity/room1`, but not messages sent to `sensor/temperature/room2` or `sensor/humidity/room2`.

## Reserved Topics

Except for the topics listed here, any topic that begins with \$ is considered reserved and is not supported for publishing or subscribing. Attempt to publish or subscribe to a topic that begins with \$ results in a terminated connection.

### Event Topics

| Topic                                                        | Allowed Operations | Description                                                                                                                 |
|--------------------------------------------------------------|--------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <code>\$aws/events/presence/connected/<i>clientId</i></code> | Subscribe          | AWS IoT publishes to this topic when an MQTT client with the specified client ID connects to AWS IoT. For more information, |

| Topic                                                    | Allowed Operations | Description                                                                                                                                                                                       |
|----------------------------------------------------------|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                          |                    | see <a href="#">Connect/Disconnect Events (p. 693)</a> .                                                                                                                                          |
| \$aws/events/presence/disconnected/ <i>clientId</i>      | Subscribe          | AWS IoT publishes to this topic when an MQTT client with the specified client ID disconnects to AWS IoT. For more information, see <a href="#">Connect/Disconnect Events (p. 693)</a> .           |
| \$aws/events/subscriptions/subscribed/ <i>clientId</i>   | 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 <a href="#">Subscribe/Unsubscribe Events (p. 696)</a> .   |
| \$aws/events/subscriptions/unsubscribed/ <i>clientId</i> | 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 <a href="#">Subscribe/Unsubscribe Events (p. 696)</a> . |

## Rule Topics

| Topic                        | Allowed Operations | Description                                                                                                                                                                   |
|------------------------------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$aws/rules/ <i>ruleName</i> | Publish            | A device or an application publishes to this topic to trigger rules directly. For more information, see <a href="#">Reducing Messaging Costs with Basic Ingest (p. 293)</a> . |

## Thing Shadow Topics

| Topic                                                    | Allowed Operations | Description                                                                                                                                   |
|----------------------------------------------------------|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| \$aws/things/< <i>thingName</i> >/shadow/delete          | Publish/Subscribe  | A device or an application publishes to this topic to delete a shadow. For more information, see <a href="#">/delete</a> .                    |
| \$aws/things/< <i>thingName</i> >/shadow/delete/accepted | Subscribe          | The Device Shadow service sends messages to this topic when a shadow is deleted. For more information, see <a href="#">/delete/accepted</a> . |
| \$aws/things/< <i>thingName</i> >/shadow/delete/rejected | Subscribe          | The Device Shadow service sends messages to this topic                                                                                        |

| Topic                                           | Allowed Operations | Description                                                                                                                                                                                           |
|-------------------------------------------------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                 |                    | when a request to delete a shadow is rejected. For more information, see <a href="#">/delete/rejected</a> .                                                                                           |
| \$aws/things/<thingName>/shadow/get             | Publish/Subscribe  | An application or a thing publishes an empty message to this topic to get a shadow. For more information, see <a href="#">Shadow MQTT Topics</a> .                                                    |
| \$aws/things/<thingName>/shadow/get/accepted    | Subscribe          | The Device Shadow service sends messages to this topic when a request for a shadow is made successfully. For more information, see <a href="#">/get/accepted</a> .                                    |
| \$aws/things/<thingName>/shadow/get/rejected    | Subscribe          | The Device Shadow service sends messages to this topic when a request for a shadow is rejected. For more information, see <a href="#">/get/rejected</a> .                                             |
| \$aws/things/<thingName>/shadow/update          | Publish/Subscribe  | A thing or application publishes to this topic to update a shadow. For more information, see <a href="#">/update</a> .                                                                                |
| \$aws/things/<thingName>/shadow/update/accepted | Subscribe          | The Device Shadow service sends messages to this topic when an update is successfully made to a shadow. For more information, see <a href="#">/update/accepted</a> .                                  |
| \$aws/things/<thingName>/shadow/update/rejected | Subscribe          | The Device Shadow service sends messages to this topic when an update to a shadow is rejected. For more information, see <a href="#">/update/rejected</a> .                                           |
| \$aws/things/<thingName>/shadow/update/delta    | Subscribe          | The Device Shadow service sends messages to this topic when a difference is detected between the reported and desired sections of a shadow. For more information, see <a href="#">/update/delta</a> . |

| Topic                                            | Allowed Operations | Description                                                                                                                                                                |
|--------------------------------------------------|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$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 <a href="#">/update/documents</a> . |

## Job Topics

| Topic                                             | Allowed Operations | Description                                                                                                                                                                        |
|---------------------------------------------------|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$aws/things/<thingName>/jobs/get                 | Publish            | Devices publish a message to this topic to make a GetPendingJobExecutions request. For more information, see <a href="#">Using the AWS IoT Jobs APIs</a> .                         |
| \$aws/things/<thingName>/jobs/get/accepted        | Subscribe          | Devices subscribe to this topic to receive successful responses from a GetPendingJobExecutions request. For more information, see <a href="#">Using the AWS IoT Jobs APIs</a> .    |
| \$aws/things/<thingName>/jobs/get/rejected        | Subscribe          | Devices subscribe to this topic when a GetPendingJobExecutions request is rejected. For more information, see <a href="#">Using the AWS IoT Jobs APIs</a> .                        |
| \$aws/things/<thingName>/jobs/start-next          | Publish            | Devices publish a message to this topic to make a StartNextPendingJobExecution request. For more information, see <a href="#">Using the AWS IoT Jobs APIs</a> .                    |
| \$aws/things/<thingName>/jobs/start-next/accepted | Subscribe          | Devices subscribe to this topic to receive successful responses to a StartNextPendingJobExecution request. For more information, see <a href="#">Using the AWS IoT Jobs APIs</a> . |
| \$aws/things/<thingName>/jobs/start-next/rejected | Subscribe          | Devices subscribe to this topic when a StartNextPendingJobExecution request is rejected. For more information, see <a href="#">Using the AWS IoT Jobs APIs</a> .                   |

| Topic                                                 | Allowed Operations | Description                                                                                                                                                                                                                                                                                                                                         |
|-------------------------------------------------------|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$aws/things/<thingName>/jobs/<jobId>/get             | Publish            | Devices publish a message to this topic to make a <code>DescribeJobExecution</code> request. For more information, see <a href="#">Using the AWS IoT Jobs APIs</a> .                                                                                                                                                                                |
| \$aws/things/<thingName>/jobs/<jobId>/get/accepted    | Subscribe          | Devices subscribe to this topic to receive successful responses to a <code>DescribeJobExecution</code> request. For more information, see <a href="#">Using the AWS IoT Jobs APIs</a> .                                                                                                                                                             |
| \$aws/things/<thingName>/jobs/<jobId>/get/rejected    | Subscribe          | Devices subscribe to this topic when a <code>DescribeJobExecution</code> request is rejected. For more information, see <a href="#">Using the AWS IoT Jobs APIs</a> .                                                                                                                                                                               |
| \$aws/things/<thingName>/jobs/<jobId>/update          | Publish            | Devices publish a message to this topic to make an <code>UpdateJobExecution</code> request. For more information, see <a href="#">Using the AWS IoT Jobs APIs</a> .                                                                                                                                                                                 |
| \$aws/things/<thingName>/jobs/<jobId>/update/accepted | Subscribe          | <p>Devices subscribe to this topic to receive successful responses to an <code>UpdateJobExecution</code> request. For more information, see <a href="#">Using the AWS IoT Jobs APIs</a>.</p> <p><b>Note</b><br/>Only the device that publishes to \$aws/things/&lt;thingName&gt;/jobs/&lt;jobId&gt;/update will receive messages on this topic.</p> |

| Topic                                                 | Allowed Operations | Description                                                                                                                                                                                                                                                                                                                                    |
|-------------------------------------------------------|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$aws/things/<thingName>/jobs/<jobId>/update/rejected | Subscribe          | <p>Devices subscribe to this topic when an <code>UpdateJobExecution</code> request is rejected. For more information, see <a href="#">Using the AWS IoT Jobs APIs</a>.</p> <p><b>Note</b><br/>Only the device that publishes to <code>\$aws/things/&lt;thingName&gt;/jobs/&lt;jobId&gt;/update</code> will receive messages on this topic.</p> |
| \$aws/things/<thingName>/jobs/notify                  | Subscribe          | <p>Devices subscribe to this topic to receive notifications when a job execution is added or removed to the list of pending executions for a thing. For more information, see <a href="#">Using the AWS IoT Jobs APIs</a>.</p>                                                                                                                 |
| \$aws/things/<thingName>/jobs/notify-next             | Subscribe          | <p>Devices subscribe to this topic to receive notifications when the next pending job execution for the thing is changed. For more information, see <a href="#">Using the AWS IoT Jobs APIs</a>.</p>                                                                                                                                           |
| \$aws/events/job/<jobId>/completed                    | Subscribe          | <p>The Jobs service publishes an event on this topic when a job completes. For more information, see <a href="#">Job Events</a>.</p>                                                                                                                                                                                                           |
| \$aws/events/job/<jobId>/canceled                     | Subscribe          | <p>The Jobs service publishes an event on this topic when a job is canceled. For more information, see <a href="#">Job Events</a>.</p>                                                                                                                                                                                                         |
| \$aws/events/job/<jobId>/deleted                      | Subscribe          | <p>The Jobs service publishes an event on this topic when a job is deleted. For more information, see <a href="#">Job Events</a>.</p>                                                                                                                                                                                                          |
| \$aws/events/job/<jobId>/cancellation_in_progress     | Subscribe          | <p>The Jobs service publishes an event on this topic when a job cancellation begins. For more information, see <a href="#">Job Events</a>.</p>                                                                                                                                                                                                 |

| Topic                                                  | Allowed Operations | Description                                                                                                                                |
|--------------------------------------------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| \$aws/events/job/< <i>jobId</i> >/deletion_in_progress | Subscribe          | The Jobs service publishes an event on this topic when a job deletion begins. For more information, see <a href="#">Job Events</a> .       |
| \$aws/events/jobExecution/< <i>jobId</i> >/succeeded   | Subscribe          | The Jobs service publishes an event on this topic when job execution succeeds. For more information, see <a href="#">Job Events</a> .      |
| \$aws/events/jobExecution/< <i>jobId</i> >/failed      | Subscribe          | The Jobs service publishes an event on this topic when a job execution fails. For more information, see <a href="#">Job Events</a> .       |
| \$aws/events/jobExecution/< <i>jobId</i> >/rejected    | Subscribe          | The Jobs service publishes an event on this topic when a job execution is rejected. For more information, see <a href="#">Job Events</a> . |
| \$aws/events/jobExecution/< <i>jobId</i> >/canceled    | Subscribe          | The Jobs service publishes an event on this topic when a job execution is canceled. For more information, see <a href="#">Job Events</a> . |
| \$aws/events/jobExecution/< <i>jobId</i> >/timed_out   | Subscribe          | The Jobs service publishes an event on this topic when a job execution times out. For more information, see <a href="#">Job Events</a> .   |
| \$aws/events/jobExecution/< <i>jobId</i> >/removed     | Subscribe          | The Jobs service publishes an event on this topic when a job execution is removed. For more information, see <a href="#">Job Events</a> .  |
| \$aws/events/jobExecution/< <i>jobId</i> >/deleted     | Subscribe          | The Jobs service publishes an event on this topic when a job execution is deleted. For more information, see <a href="#">Job Events</a> .  |

## Fleet Provisioning Topics

| Topic                                               | Allowed Operations | Description                                                                                                                                                                        |
|-----------------------------------------------------|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$aws/events/presence/connected/< <i>clientId</i> > | Subscribe          | AWS IoT publishes to this topic when an MQTT client with the specified client ID connects to AWS IoT. For more information, see <a href="#">Connect/Disconnect Events (p. 693)</a> |

| Topic                                                                   | Allowed Operations | Description                                                                                                |
|-------------------------------------------------------------------------|--------------------|------------------------------------------------------------------------------------------------------------|
| \$aws/certificates/create/cbor                                          | Publish            | You publish to this topic to call the CreateKeysAndCertificate MQTT API.                                   |
| \$aws/certificates/create/json                                          | Publish            | You publish to this topic to call the CreateKeysAndCertificate MQTT API.                                   |
| \$aws/provisioning-templates/<br><templateName>/provision/cbor          | Publish            | You publish to this topic to call the RegisterThing MQTT API.                                              |
| \$aws/provisioning-templates/<br><templateName>/provision/json          | Publish            | You publish to this topic to call the RegisterThing MQTT API.                                              |
| \$aws/certificates/create/cbor/accepted                                 | Subscribe          | AWS IoT publishes to this topic when a call is successfully made to the CreateKeysAndCertificate MQTT API. |
| \$aws/certificates/create/cbor/rejected                                 | Subscribe          | AWS IoT publishes to this topic when a call to the CreateKeysAndCertificate MQTT API fails.                |
| \$aws/certificates/create/json/accepted                                 | Subscribe          | AWS IoT publishes to this topic when a call is made successfully to the CreateKeysAndCertificate MQTT API. |
| \$aws/certificates/create/json/rejected                                 | Subscribe          | AWS IoT publishes to this topic when a call to the CreateKeysAndCertificate MQTT API fails.                |
| \$aws/provisioning-templates/<br><templateName>/provision/cbor/accepted | Subscribe          | AWS IoT publishes to this topic when a call is made successfully to the RegisterThing MQTT API.            |
| \$aws/provisioning-templates/<br><templateName>/provision/cbor/rejected | Subscribe          | AWS IoT publishes to this topic when a call to the RegisterThing MQTT API fails.                           |
| \$aws/provisioning-templates/<br><templateName>/provision/json/accepted | Subscribe          | AWS IoT publishes to this topic when a call is made successfully to the RegisterThing MQTT API.            |

| Topic                                                                   | Allowed Operations | Description                                                                      |
|-------------------------------------------------------------------------|--------------------|----------------------------------------------------------------------------------|
| \$aws/provisioning-templates/<br><templateName>/provision/json/rejected | Subscribe          | AWS IoT publishes to this topic when a call to the RegisterThing MQTT API fails. |

## Protocols

The message broker supports clients that use the MQTT protocol to publish and subscribe to messages and the HTTPS protocol to publish messages. Both protocols are supported through IP version 4 and IP version 6. The message broker also supports the MQTT protocol over the WebSocket protocol. The way in which a client can connect to the message broker depends on the protocol used.

## Protocols, Port Mappings, and Authentication

The following table lists the protocols supported by AWS IoT and the authentication methods and ports used by each.

### Protocols, Authentication, and Port Mappings

| Protocol            | Authentication           | Port             | ALPN Protocol Name |
|---------------------|--------------------------|------------------|--------------------|
| MQTT over WebSocket | Signature Version 4      | 443              | N/A                |
| MQTT                | X.509 client certificate | 443 <sup>†</sup> | x-amzn-mqtt-ca     |
| MQTT                | X.509 client certificate | 8883             | N/A                |
| HTTPS               | Signature Version 4      | 443              | N/A                |
| HTTPS               | X.509 client certificate | 443 <sup>†</sup> | x-amzn-http-ca     |
| HTTPS               | X.509 client certificate | 8443             | N/A                |

<sup>†</sup>Clients that connect on port 443 with X.509 client certificate authentication must implement the [Application Layer Protocol Negotiation \(ALPN\)](#) TLS extension and use the [ALPN protocol name](#) listed in the ALPN ProtocolNameList sent by the client as part of the `ClientHello` message. Clients must also send the [Server Name Indication \(SNI\) TLS extension](#) to prevent their connections from being refused. For more information, see [Transport Security in AWS IoT](#).

## MQTT

[MQTT](#) is a lightweight and widely adopted messaging protocol designed for constrained devices. For more information, see the [MQTT v3.1.1 specification](#).

The AWS IoT message broker implementation is based on MQTT version 3.1.1, but it deviates from the specification as follows:

- AWS IoT Core supports MQTT Quality of Service (QoS) levels 0 and 1 only. AWS IoT Core does not support publishing or subscribing with QoS level 2. When QoS level 2 is requested, the AWS IoT message broker does not send a PUBACK or SUBACK.
- In AWS IoT Core, subscribing to a topic with QoS level 0 means a message is 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.

- 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.
- 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 Core 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 cannot be connected concurrently to the message broker. When a client connects to the message broker using a client ID that another client is using, the new client connection is accepted and the previously connected client is 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.

You connect to AWS IoT Core over MQTT by using one of the [AWS IoT Device and Mobile SDKs \(p. 706\)](#). For an example of how to connect to AWS IoT Core with MQTT, see the `basicPubSub` sample in the [Python SDK](#). The other [AWS IoT SDKs](#) have similar sample apps.

For more information about authentication and port mappings for MQTT messages, see [Protocols, Port Mappings, and Authentication \(p. 253\)](#).

## MQTT Persistent Sessions

A persistent session represents an ongoing connection to an MQTT message broker. When a client connects to the AWS IoT message broker using a persistent session, the message broker saves all subscriptions the client makes during the connection. When the client disconnects, the message broker stores unacknowledged QoS 1 messages and new QoS 1 messages published to topics to which the client is subscribed. When the client reconnects to the persistent session, all subscriptions are reinstated and all stored messages are sent to the client at a maximum rate of 10 messages per second.

You create an MQTT persistent session by sending a CONNECT message, setting the `cleanSession` flag to 0. If no session exists for the client sending the CONNECT message, a new persistent session is created. If a session already exists for the client, it is resumed.

After the client joins the session, it can continue to publish messages and subscribe to topic filters without any additional flags on each operation. The following conditions describe how persistent sessions begin and end.

- A persistent session ends and can't be resumed when the client sends a CONNECT message that sets the `cleanSession` flag to 1.
- By default, a persistent session expires one hour after the message broker detects that a client has disconnected. You can configure this time interval.
- When a client reconnects after the session has expired and sets the `cleanSession` flag to 0, the service creates a new persistent session. Subscriptions and messages from the previous session are discarded.

Devices use the `sessionPresent` attribute in the connection acknowledged (CONNACK) message to determine if a persistent session is present. If `sessionPresent` is set to 1, a persistent session is present and stored messages are delivered to the client. This starts immediately after the device receives the CONNACK. There is no need to resubscribe. If `sessionPresent` is set to 0, no persistent session is present and the client must resubscribe to its topic filters.

Persistent sessions have a default expiry period of one hour. The expiry period begins when the message broker detects that a client disconnects (MQTT disconnect or timeout). The persistent session expiry period can be increased through the standard limit increase process. If a client has not resumed its session within the expiry period, the session is terminated and any associated stored messages are discarded. The expiry period is approximate. Sessions might be persisted for up to 30 minutes longer, but not less than the configured duration. For more information, see [AWS Service Quotas](#). Any messages stored for persistent sessions are billed at the standard messaging rate. For more information, see [AWS IoT Core Pricing](#).

## MQTT over the WebSocket Protocol

AWS IoT Core supports MQTT over the [WebSocket](#) protocol to enable browser-based and remote applications to send and receive data from AWS IoT Core-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 Core endpoint. You can use the AWS IoT Core CLI [describe-endpoint](#) command to find this endpoint.

region

The AWS Region of your AWS account.

mqtt

Specifies you are sending MQTT messages over the WebSocket protocol.

When the server responds, the client sends an upgrade request to indicate to the server it communicates 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 to generate a Signature Version 4 request.

For more information about using the WebSocket protocol in JavaScript, see [WebSocket](#) in *The Modern JavaScript Tutorial*.

```
/**  
 * 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 that you use one of the AWS IoT Core Device SDKs to connect your device to AWS IoT Core when making a WebSocket connection. The following AWS IoT Core Device SDKs support WebSocket-based MQTT connections to AWS IoT Core:

- [Node.js](#)
- [iOS](#)
- [Android](#)

For a reference implementation for connecting a web application to AWS IoT Core using MQTT over the WebSocket protocol, see [AWS Labs WebSocket sample](#) on the GitHub website.

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 Signature Version 4. Some MQTT clients, such as [Eclipse Paho for JavaScript](#), support the WebSocket protocol natively.

## HTTP

Clients can publish messages by making requests to the REST API using the HTTP 1.0 or 1.1 protocols. For the authentication and port mappings used by HTTP requests, see [Protocols, Port Mappings, and Authentication \(p. 253\)](#).

### HTTP URL

Publishing clients make POST requests to a client-specific endpoint and a topic-specific URL:  
`https://<AWS IoT endpoint>/topics/<url_encoded_topic_name>?qos=1".`

- `<AWS IoT endpoint>` is the REST API endpoint. You can find the endpoint in the AWS IoT console on the thing's details page or on the client by using the AWS IoT CLI command:

**aws iot describe-endpoint**

The endpoint should look something like this: `a3qj468xinsffp-ats.iot.us-west-2.amazonaws.com`

- `<url_encoded_topic_name>` is the full [topic name](#) of the message being sent.

### curl Example

You can use [curl](#) to emulate a client sending a message to AWS IoT Core.

#### To use curl to send a message from an AWS IoT Core client device

1. Check the curl version.
  - a. On your client, run this command at a command prompt.

**curl --help**

In the help text, look for the TLS options. You should see the `--tlsv1.2` option.

- b. If you see the `--tlsv1.2` option, continue.
  - c. If you don't see the `--tlsv1.2` option or you get a `command not found` error, update or install curl on your client before you continue.
2. Install the certificates on your client.

Copy the certificate files that you created when you registered your client (thing) in the AWS IoT console. Make sure you have these three certificate files on your client before you continue.

- The client's private key file (`private.key` in this example).
- The client's certificate file (`certificate.pem.crt` in this example).
- The CA certificate file (`AmazonRootCA1.pem` in this example).

3. Create the curl command line.

```
curl --tlsv1.2 --cacert AmazonRootCA1.pem --cert certificate.pem.crt --key private.key
-X POST -d "{\"message\": \"Hello, world\" }" "https://<AWS IoT Endpoint>:8443/
topics/topic?qos=1">
```

`--tlsv1.2`

Use TLS 1.2 (SSL).

`--cacert <filename>`

The file name and path, if necessary, of the CA certificate to verify the peer.

`--cert <filename>`

The client's certificate file name and path, if necessary.

`--key <filename>`

The client's private key file name and path, if necessary.

`-X POST`

The type of HTTP request (in this case, POST).

`-d <data>`

The HTTP POST data you want to publish. In this case, it's a JSON string, with the internal quotation marks escaped with the backslash character (\).

"<https://<AWS IoT endpoint>:8443/topics/topic?qos=1>"

The URL of your client's REST API endpoint, followed by the HTTPS port, :8443, which is then followed by the keyword, /topics/ and the topic name, topic, in this case. Specify the quality of service as the query parameter, ?qos=1.

4. Open the MQTT test client in the console.

Follow the instructions in [View Device MQTT Messages with the AWS IoT MQTT Client](#) and configure the console to subscribe to messages with the topic name of `topic`, or use the wildcard topic filter of `#`.

5. Test the command.

While monitoring the topic in the test client of the AWS IoT console, go to your client and issue the curl command line you created in step 3. You should see your client's messages in the console.

# 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.
- Send message data to an AWS IoT Analytics channel.
- Start execution of a Step Functions state machine.
- Send message data to an AWS IoT Events input.
- Send message data to an asset property in AWS IoT SiteWise.
- Send message data to a web application or service.

Your rules can use MQTT messages that pass through the publish/subscribe [Message Broker for AWS IoT \(p. 244\)](#) or, using the [Basic Ingest \(p. 293\)](#) feature, you can securely send device data to the AWS services listed above without incurring [messing costs](#). (The [Basic Ingest \(p. 293\)](#) feature optimizes data flow by removing the publish/subscribe message broker from the ingestion path, so it is more cost effective while keeping the security and data processing features of AWS IoT.)

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.

## 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 named `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.json
```

For more information about how to grant access to AWS services in policies for AWS IoT, see [Creating an AWS IoT Rule \(p. 262\)](#).

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",  
        "LastUpdated": "2015-09-30T19:31:18.620Z",  
        "IsAttachmentEnabled": true,  
        "AttachmentCount": 0,  
        "Arn": "arn:aws:iam::123456789012:policy/my-iot-policy"  
    }  
}
```

```
        "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 you create 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.

**Note**

We do not recommend the use of personally identifiable information in your rule names.

#### Optional description

A textual description of the rule.

**Note**

We do not recommend the use of personally identifiable information in your rule descriptions.

#### 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 \(p. 294\)](#).

#### 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. For more information, see [SQL Versions \(p. 346\)](#).

#### 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 Elasticsearch Service:

```

{
    "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"
        }
    ]
}
```

The following is an example payload file with a rule that starts an execution of a Step Functions state machine.

```
{
    "sql": "expression",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
        "stepFunctions": {
            "stateMachineName": "myCoolStateMachine",
            "executionNamePrefix": "coolRunning",
            "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
        }
    ]
}
```

## 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
```

## 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.
- `cloudwatchLogs` to send data to CloudWatch Logs.
- `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.
- `http` to post data to an HTTPS endpoint.
- `iotAnalytics` to send data to an AWS IoT Analytics channel.
- `iotEvents` to send data to an AWS IoT Events input.
- `iotSiteWise` to send data to asset properties in AWS IoT SiteWise.
- `kinesis` to write data to a Kinesis stream.
- `lambda` to invoke a Lambda function.
- `republish` to republish the message on another MQTT topic.
- `s3` to write data to an Amazon S3 bucket.
- `salesforce` to write a message to a Salesforce IoT input stream.
- `sns` to write data as a push notification.
- `sqs` to write data to an SQS queue.
- `stepFunctions` to start execution of a Step Functions state machine.

## Note

The AWS IoT rules engine might make multiple attempts to perform an action in case of intermittent errors. If all attempts fail, the message is discarded and the error is available in your CloudWatch logs. You can specify an error action for each rule that is invoked after a failure occurs. For more information, see [Error Handling \(Error Action\) \(p. 289\)](#).

Some rule actions trigger actions in services that integrate with AWS Key Management Service (AWS KMS) to support data encryption at rest. If you use a customer managed AWS KMS customer master key (CMK) to encrypt data at rest, the service must have permission to use the CMK on the caller's behalf. See the data encryption topics in the appropriate service guide to learn how to manage permissions for your customer managed CMK. For more information about CMKs and customer managed CMKs, see [AWS Key Management Service concepts](#) in the *AWS Key Management Service Developer Guide*.

Each action is described in detail.

## CloudWatch Alarm Action

### 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.

[More information \(1\)](#)

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**

Make sure 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:

```
{  
    "topicRulePayload": {  
        "sql": "SELECT * FROM 'some/topic'",  
        "ruleDisabled": false,  
        "awsIotSqlVersion": "2016-03-23",  
        "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 Logs Action

### CloudWatch Logs Action

The CloudWatch logs action allows you to send data to CloudWatch Logs. You can specify the CloudWatch log group to which the action sends data.

#### More information (2)

When you create an AWS IoT rule with a CloudWatch logs action, you must specify the following information:

roleArn

The IAM role that allows access to the CloudWatch log.

logGroupName

The CloudWatch log group to which the action sends data.

The following JSON example shows how to define a CloudWatch logs action in an AWS IoT rule:

```
{  
    "topicRulePayload": {  
        "sql": "SELECT * FROM 'some/topic'",  
        "ruleDisabled": false,  
        "awsIotSqlVersion": "2016-03-23",  
        "actions": [  
            {"cloudwatchLogs": {  
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw",  
                "logGroupName": "IotLogs"  
            }}  
        ]  
    }  
}
```

For more information, see [Getting Started with CloudWatch Logs](#). If you use a customer managed AWS KMS CMK to encrypt log data in CloudWatch Logs, the service must have permission to use the CMK on the caller's behalf. For more information, see [Encrypt Log Data in CloudWatch Logs Using AWS KMS](#) in the *Amazon CloudWatch Logs User Guide*.

## CloudWatch Metric Action

### 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.

More information (3)

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

Make sure that 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:

```
{
```

```
"topicRulePayload": {  
    "sql": "SELECT * FROM 'some/topic'",  
    "ruleDisabled": false,  
    "awsIotSqlVersion": "2016-03-23",  
    "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

### DynamoDB Action

The `dynamoDB` action allows you to write all or part of an MQTT message to a DynamoDB table.

[More information \(4\)](#)

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.

Make sure that 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:

```
{  
    "topicRulePayload": {  
        "ruleDisabled": false,  
        "sql": "SELECT * AS message FROM 'some/topic'",  
        "description": "A test Dynamo DB rule",  
        "awsIotSqlVersion": "2016-03-23",  
        "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](#). If you use a customer managed AWS KMS CMK to encrypt data at rest in DynamoDB, the service must have permission to use the CMK on the caller's behalf. For more information, see [Customer Managed CMK](#) in the [Amazon DynamoDB Getting Started Guide](#).

## DynamoDBv2 Action

### 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.

#### More information (5)

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**

Make sure that 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:

```
{  
    "topicRulePayload": {  
        "sql": "SELECT * AS message FROM 'some/topic'",  
        "ruleDisabled": false,  
        "description": "A test DynamoDBv2 rule",  
        "awsIotSqlVersion": "2016-03-23",  
        "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](#). If you use a customer managed AWS KMS CMK to encrypt data at rest in DynamoDB, the service must have permission to use the CMK on the caller's behalf. For more information, see [Customer Managed CMK](#) in the [Amazon DynamoDB Getting Started Guide](#).

## Elasticsearch Action

### Elasticsearch Action

The `elasticsearch` action allows you to write data from MQTT messages to an Amazon Elasticsearch Service domain. Data in Elasticsearch can then be queried and visualized by using tools like Kibana.

#### More information (6)

When you create an AWS IoT rule with an `elasticsearch` action, you must specify the following information:

endpoint

The endpoint of your Amazon Elasticsearch Service domain.

index

The Elasticsearch index where you want to store your data.

type

The type of document you are storing.

id

The unique identifier for each document.

**Note**

Make sure that 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:

```
{  
  "topicRulePayload":{  
    "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()}"  
      }  
    }  
  }  
}
```

For more information, see the [Amazon Elasticsearch Service Developer Guide](#). If you use a customer managed AWS KMS CMK to encrypt data at rest in Elasticsearch, the service must have permission to use the CMK on the caller's behalf. For more information, see [Encryption of Data at Rest for Amazon Elasticsearch Service](#) in the [Amazon Elasticsearch Service Developer Guide](#).

## Firehose Action

### Firehose Action

A firehose action sends data from an MQTT message that triggered the rule to a Kinesis Data Firehose stream.

#### More information (7)

When you create 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 Kinesis Data Firehose stream. Valid values are: '\n' (newline), '\t' (tab), '\r\n' (Windows newline), ',' (comma).

**Note**

Make sure that 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:

```
{  
    "topicRulePayload": {  
        "sql": "SELECT * FROM 'some/topic'",  
        "ruleDisabled": false,  
        "awsIotSqlVersion": "2016-03-23",  
        "actions": [  
            {"  
                "firehose": {  
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_firehose",  
                    "deliveryStreamName": "my_firehose_stream"  
                }  
            }  
        ]  
    }  
}
```

For more information, see the [Amazon Kinesis Data Firehose Developer Guide](#). If you use Kinesis Data Firehose to send data to an Amazon S3 bucket and you use a customer managed AWS KMS CMK to encrypt data at rest in Amazon S3, Kinesis Data Firehose must have access to your bucket and permission to use the CMK on the caller's behalf. For more information, see [Grant Kinesis Data Firehose Access to an Amazon S3 Destination](#) in the [Amazon Kinesis Data Firehose Developer Guide](#).

## HTTP Action

### HTTP Action

The `http` action sends data from an MQTT message that triggered the rule to your web applications or services for further processing, without writing any code. The endpoint you send data to must be verified before the rules engine can use it.

#### More information (8)

When you create a rule with an `http` action, you must specify the following information:

##### url

The HTTPS URL where the message is sent by HTTP POST. You can use substitution templates in the URL.

##### confirmationUrl

Optional. If specified, AWS IoT uses the confirmation URL to create a matching topic rule destination. You must enable the topic rule destination before using it in an `http` action. For more information, see [Working with Topic Rule Destinations \(p. 291\)](#). If you use substitution templates, you must manually create topic rule destinations before the `http` action can be used. `confirmationUrl` must be a prefix of `url`.

The relationship between `url` and `confirmationUrl` is described by the following:

- If `url` is hardcoded and `confirmationUrl` is not provided, we implicitly treat the `url` field as the `confirmationUrl`. AWS IoT creates a topic rule destination for `url`.
- If `url` and `confirmationUrl` are hardcoded, `url` must begin with `confirmationUrl`. AWS IoT creates a topic rule destination for `confirmationUrl`.
- If `url` contains a substitution template, you must specify `confirmationUrl` and `url` must begin with `confirmationUrl`. If `confirmationUrl` contains substitution templates,

you must manually create topic rule destinations before the `http` action can be used. If `confirmationUrl` does not contain substitution templates, AWS IoT creates a topic rule destination for `confirmationUrl`.

**headers**

Optional. Any headers you want to include in HTTP requests.

**key**

The key of the header. Substitution templates are not supported.

**value**

The value of the header. Substitution templates are supported.

**auth**

Optional. The authentication used by the rules engine to connect to the endpoint URL specified in the `url` argument. Currently, Signature Version 4 is the only supported authentication type. For more information, see [HTTP Authorization](#).

The following JSON example shows how to create an AWS IoT rule with an `http` action:

```
{  
    "topicRulePayload": {  
        "sql": "select * from 'a/b'",  
        "awsIotSqlVersion": "2016-03-23",  
        "ruleDisabled": false,  
        "actions": [{  
            "http": {  
                "url": "https://www.example.com/subpath",  
                "confirmationUrl": "https://www.example.com",  
                "headers": [{  
                    "key": "static_header_key",  
                    "value": "static_header_value"  
                },  
                {  
                    "key": "substitutable_header_key",  
                    "value": "${value_from_payload}"  
                }]  
            }  
        }]  
    }  
}
```

The default content type is `application/json` when the payload is in JSON format. Otherwise, it is `application/octet-stream`. You can overwrite it by specifying the exact content type in the header with the key `content-type` (case insensitive).

#### HTTP action retry logic

The AWS IoT rules engine retries `http` actions according to these rules:

- The rules engine tries to send a message at least once.
- The rules engine retries at most twice. The maximum number of tries is three.
- The rules engine does not attempt a retry if:
  - The previous try provided a response larger than 16384 bytes.
  - The downstream web service or application closes the TCP connection after the try.
  - The total time to complete a request with retries exceeded the request timeout limit.
  - The request returns an HTTP status code other than 429, 500-599.

**Note**

Standard data transfer costs apply to retries.

## IoT Analytics Action

### IoT Analytics Action

An `iotAnalytics` action sends data from the MQTT message that triggered the rule to an AWS IoT Analytics channel.

More information (9)

When you create a rule with an `iotAnalytics` action, you must specify the following information:

`channelName`

The name of the AWS IoT Analytics channel to which to write the data.

`roleArn`

The IAM role that allows access to the AWS IoT Analytics channel.

The policy attached to the role you specify should look like this:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "iotanalytics:BatchPutMessage",  
            "Resource": [  
                "arn:aws:iotanalytics:us-west-2:account-id:channel/mychannel"  
            ]  
        }  
    ]  
}
```

and have a trust relationship that looks like this:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "iot.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole",  
        }  
    ]  
}
```

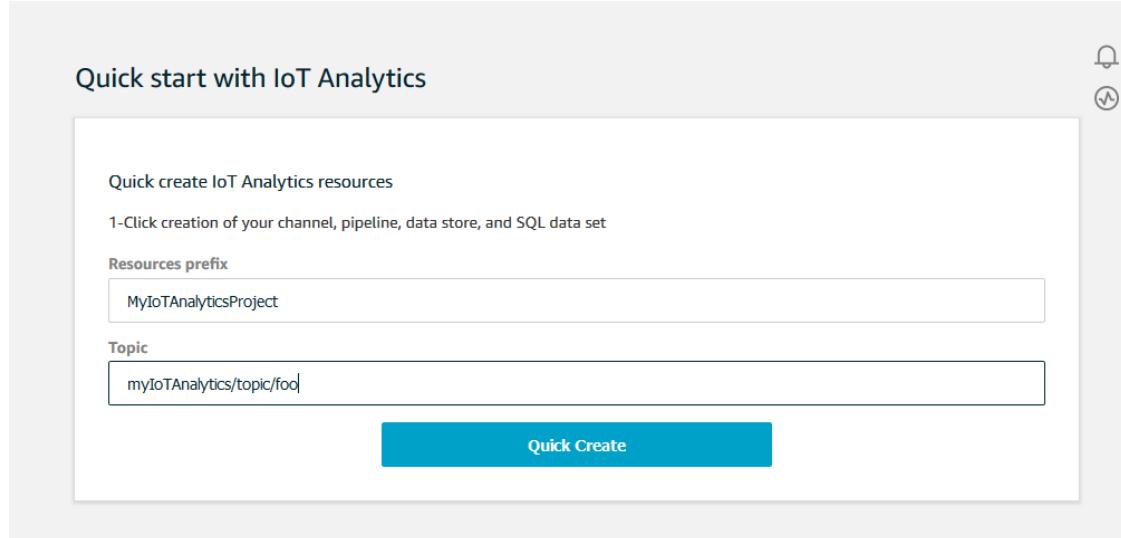
The following JSON example shows how to create an AWS IoT rule with an `iotAnalytics` action:

```
{  
    "topicRulePayload": {  
        "sql": "SELECT * FROM 'some/topic'",  
        "ruleDisabled": false,  
        "awsIotSqlVersion": "2016-03-23",  
        "actions": [  
    }
```

```
        "iotAnalytics": {  
            "channelName": "mychannel",  
            "roleArn": "arn:aws:iam::123456789012:role/analyticsRole",  
        }  
    }]  
}
```

For more information, see the [AWS IoT Analytics User Guide](#).

The AWS IoT Analytics console also has a **Quick start** feature that allows you to create a channel, data store, pipeline, and data store with one click. Look for this page when you open the AWS IoT Analytics console:



## IoT Events Action

### IoT Events Action

An `iotEvents` action sends data from the MQTT message that triggered the rule to an AWS IoT Events input.

#### More information (10)

When you create a rule with a `iotEvents` action, you must specify the following information:

##### `inputName`

The name of the AWS IoT Events input.

##### `messageId`

Optional. Use this to ensure that only one input (message) with a given `messageId` is processed by an AWS IoT Events detector.

##### `roleArn`

The ARN of the role that grants AWS IoT permission to send an input to an AWS IoT Events detector. ("Action": "iotevents:BatchPutMessage").

Here is an example trust policy that should be attached to the role:

```
{
```

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

The following JSON example shows how to create an AWS IoT rule with an `iotEvents` action:

```
{
    "topicRulePayload": {
        "sql": "expression",
        "ruleDisabled": false,
        "description": "An AWS IoT Events test rule",
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotEvents": {
                    "inputName": "MyIoTEventsInput",
                    "messageId": "1234567890",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_events"
                }
            }
        ]
    }
}
```

For more information, see the [AWS IoT Events Developer Guide](#).

## IoT SiteWise Action

### IoT SiteWise Action

An `iotSiteWise` action sends data from the MQTT message that triggered the rule to asset properties in AWS IoT SiteWise.

#### Note

When you send data to AWS IoT SiteWise with this action, your data must meet the requirements of the `BatchPutAssetPropertyValue` action. For more information, see [BatchPutAssetPropertyValue](#) in the [AWS IoT SiteWise API Reference](#).

You can follow a tutorial that shows you how to ingest data from AWS IoT things. For more information, see [Ingesting Data to AWS IoT SiteWise from AWS IoT Things](#) in the [AWS IoT SiteWise User Guide](#).

For more information about troubleshooting this rule, see [Troubleshooting an AWS IoT SiteWise Rule Action](#) in the [AWS IoT SiteWise User Guide](#).

More information (11)

When creating a rule with an `iotSiteWise` action, you must specify the following information:

`putAssetPropertyValueEntries`

A list of asset property value entries that each contain the following information:  
`entryId`

Optional. A unique identifier for this entry. You can define the `entryId` to better track which message caused an error in case of failure. Accepts substitution templates. Defaults to a new UUID.

propertyAlias

The property alias associated with your asset property. You must specify either a `propertyAlias` or both an `assetId` and a `propertyId`. Accepts substitution templates. For more information about property aliases, see [Mapping Industrial Data Streams to Asset Properties](#) in the *AWS IoT SiteWise User Guide*.

assetId

The ID of the AWS IoT SiteWise asset. You must specify either a `propertyAlias` or both an `assetId` and a `propertyId`. Accepts substitution templates.

propertyId

The ID of the asset's property. You must specify either a `propertyAlias` or both an `assetId` and a `propertyId`. Accepts substitution templates.

propertyValues

A list of property values to insert that each contain timestamp, quality, and value (TQV) in the following format:

timestamp

A timestamp structure that contains the following information:

timeInSeconds

A string that contains the time in seconds in Unix epoch time. Accepts substitution templates. If your message payload doesn't have a timestamp, you can use [timestamp\(\)](#) (p. 340), which returns the current time in milliseconds. To convert that time to seconds, you can use the following substitution template:  `${floor(timestamp() / 1E3)}` .

offsetInNanos

Optional. A string that contains the nanosecond time offset from the time in seconds. Accepts substitution templates. If your message payload doesn't have a timestamp, you can use [timestamp\(\)](#) (p. 340), which returns the current time in milliseconds. To calculate the nanosecond offset from that time, you can use the following substitution template:  `${(timestamp() % 1E3) * 1E6}` .

With respect to Unix epoch time, AWS IoT SiteWise accepts only entries that have a timestamp of up to 15 minutes in the past and up to 5 minutes in the future.

quality

Optional. A string that describes the quality of the value. Accepts substitution templates. Must be GOOD, BAD, or UNCERTAIN.

value

A value structure that contains one of the following value fields, depending on the asset property's data type:

booleanValue

Optional. A string that contains the boolean value of the value entry. Accepts substitution templates.

doubleValue

Optional. A string that contains the double value of the value entry. Accepts substitution templates.

`integerValue`

Optional. A string that contains the integer value of the value entry. Accepts substitution templates.

`stringValue`

Optional. The string value of the value entry. Accepts substitution templates.

`roleArn`

The ARN of the role that grants AWS IoT permission to send an asset property value to AWS IoT SiteWise. ("Action": "iotsitewise:BatchPutAssetPropertyValue").

Here is an example trust policy that should be attached to the role:

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

To improve security, you can specify an AWS IoT SiteWise asset hierarchy path in the Condition property. The following example is a trust policy that specifies an asset hierarchy path.

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

The following JSON example shows how to create an AWS IoT rule with a basic `iotSiteWise` action.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            "iotSiteWise": {
                "putAssetPropertyValueEntries": [

```

```
{
    "propertyAlias": "/some/property/alias",
    "propertyValues": [
        {
            "timestamp": {
                "timeInSeconds": "${my.payload.timeInSeconds}"
            },
            "value": {
                "integerValue": "${my.payload.value}"
            }
        }
    ],
    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sitewise"
},
}
}
```

The following JSON example shows how to create an AWS IoT rule with an `iotSiteWise` action. This example uses the topic as the property alias and the `timestamp()` function. For example, if you publish data to `/company/windfarm/3/turbine/7/rpm`, this action sends the data to the asset property with a property alias that is the same as the topic that you specified.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM '/company/windfarm/+/turbine/+/+'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotSiteWise": {
                    "putAssetPropertyValueEntries": [
                        {
                            "propertyAlias": "${topic()}",
                            "propertyValues": [
                                {
                                    "timestamp": {
  "timeInSeconds": "${floor(timestamp() / 1E3)}",
  "offsetInNanos": "${(timestamp() % 1E3) * 1E6}"
                                    },
                                    "value": {
  "doubleValue": "${my.payload.value}"
                                    }
                                }
                            ]
                        },
                        {
                            "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sitewise"
                        }
                    ],
                }
            }
        ]
    }
}
```

For more information, see the [AWS IoT SiteWise User Guide](#).

## Kinesis Action

### Kinesis Action

The `kinesis` action allows you to write data from MQTT messages into a Kinesis stream.

## More information (12)

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:

```
{  
    "topicRulePayload": {  
        "sql": "SELECT * FROM 'some/topic'",  
        "ruleDisabled": false,  
        "awsIotSqlVersion": "2016-03-23",  
        "actions": [  
            {  
                "kinesis": {  
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_kinesis",  
                    "streamName": "my_kinesis_stream",  
                    "partitionKey": "${topic()}"  
                }  
            }  
        ]  
    }  
}
```

For more information, see the [Amazon Kinesis Data Streams Developer Guide](#). If you use a customer managed AWS KMS CMK to encrypt data at rest in Kinesis Data Streams, the service must have permission to use the CMK on the caller's behalf. For more information, see [Permissions to Use User-Generated KMS Master Keys](#) in the [Amazon Kinesis Data Streams Developer Guide](#).

## Lambda Action

### Lambda Action

A `lambda` action calls a Lambda function, passing in the MQTT message that triggered the rule. Lambda functions are run asynchronously.

## More information (13)

Lambda functions are executed asynchronously.

In order for AWS IoT to call a Lambda function, you must configure a policy granting the `lambda:InvokeFunction` permission to AWS IoT. You can only invoke a Lambda function defined in the same region where your Lambda policy exists. 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. To allow AWS IoT to invoke a Lambda function, 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 you create 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:

```
{  
    "topicRulePayload": {  
        "sql": "SELECT * FROM 'some/topic'",  
        "ruleDisabled": false,  
        "awsIotSqlVersion": "2016-03-23",  
        "actions": [  
            {"lambda": {  
                "functionArn": "arn:aws:lambda:us-  
east-2:123456789012:function:myLambdaFunction"  
            }}  
        ]  
    }  
}
```

If you do not specify a version or alias for your Lambda function, the most recent version of the function is executed. You can specify a version or alias if you want to execute a specific version of

your Lambda function. To specify a version or alias, append the version or alias to the ARN of the Lambda function. For example:

```
"arn:aws:lambda:us-east-2:123456789012:function:myLambdaFunction:someAlias"
```

For more information about versioning and aliases see [AWS Lambda Function Versioning and Aliases](#). For more information about AWS Lambda, see the [AWS Lambda Developer Guide](#).

If you use a customer managed AWS KMS CMK to encrypt data at rest in AWS Lambda, the service must have permission to use the CMK on the caller's behalf. For more information, see [Encryption at Rest](#) in the [AWS Lambda Developer Guide](#).

## Republish Action

### Republish Action

The `republish` action allows you to republish the message that triggered the role to another MQTT topic.

[More information \(14\)](#)

When you create a rule with a `republish` action, you must specify the following information:

`topic`

The MQTT topic to which to republish the message. If you are republishing to a reserved topic, one that begins with \$ use \$\$ instead. For example, if you are republishing to a device shadow topic like `$aws/things/MyThing/shadow/update`, specify the topic as `$aws/things/MyThing/shadow/update`.

`roleArn`

The IAM role that allows publishing to the MQTT topic.

`qos`

Optional. The Quality of Service (QoS) level to use when republishing messages. Valid values are 0 or 1. The default value is 0.

#### Note

Make sure that the role associated with the rule has a policy granting the `iot:Publish` permission.

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'some/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "another/topic",
          "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish",
          "qos": 1
        }
      }
    ]
  }
}
```

## S3 Action

### S3 Action

An s3 action writes the data from the MQTT message that triggered the rule to an Amazon S3 bucket.

[More information \(15\)](#)

When creating an AWS IoT rule with an s3 action, you must specify the following information, except cannedacl, which is optional:

bucket

The Amazon S3 bucket to which to write data.

cannedacl

Optional. The Amazon S3 canned ACL that controls access to the object identified by the object key. For more information, including allowed values, 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.". If 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 that 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:

```
{  
    "topicRulePayload": {  
        "sql": "SELECT * FROM 'some/topic'",  
        "ruleDisabled": false,  
        "awsIotSqlVersion": "2016-03-23",  
        "actions": [{  
            "s3": {  
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",  
                "bucketName": "my-bucket",  
                "key": "${topic()}/${timestamp()}"  
                "cannedacl": "public-read"  
            }  
        }]  
    }  
}
```

For more information, see the [Amazon Simple Storage Service Developer Guide](#). If you use a customer managed AWS KMS CMK to encrypt data at rest in Amazon S3, the service must have

permission to use the CMK on the caller's behalf. For more information, see [AWS Managed CMKs and Customer Managed CMKs](#) in the *Amazon Simple Storage Service Developer Guide*.

## Salesforce Action

### Salesforce Action

A **salesforce** action sends data from the MQTT message that triggered the rule to a Salesforce IoT input stream.

[More information \(16\)](#)

When you create 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. For more information, see the [Salesforce IoT documentation](#).

**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. For more information, see the [Salesforce IoT documentation](#).

**Note**

These parameters do not support substitution.

The following JSON example shows how to create an AWS IoT rule with a **salesforce** action:

```
{  
    "topicRulePayload": {  
        "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, see the [Salesforce IoT documentation](#).

## SNS Action

### SNS Action

A **sns** action sends the data from the MQTT message that triggered the rule as an SNS push notification.

[More information \(17\)](#)

When you create 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 that 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:

```
{  
    "topicRulePayload": {  
        "sql": "SELECT * FROM 'some/topic'",  
        "ruleDisabled": false,  
        "awsIotSqlVersion": "2016-03-23",  
        "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 Simple Notification Service Developer Guide](#). If you use a customer managed AWS KMS CMK to encrypt data at rest in Amazon SNS, the service must have permission to use the CMK on the caller's behalf. For more information, see [Key Management](#) in the [Amazon Simple Notification Service Developer Guide](#).

## SQS Action

### SQS Action

An `sqs` action sends data from the MQTT message that triggered the rule to an SQS queue.

[More information \(18\)](#)

When you create 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 that 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:

```
{  
    "topicRulePayload": {  
        "sql": "SELECT * FROM 'some/topic'",  
        "ruleDisabled": false,  
        "awsIotSqlVersion": "2016-03-23",  
        "actions": [  
            {  
                "sqs": {  
                    "queueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/  
my_sqs_queue",  
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sq",  
                    "useBase64": false  
                }  
            }  
        ]  
    }  
}
```

SQS action does not support [SQS FIFO Queues](#). Because the Rules Engine is a fully distributed service, there is no guarantee of message order when the SQS action is triggered.

For more information, see the [Amazon Simple Queue Service Developer Guide](#). If you use a customer managed AWS KMS CMK to encrypt data at rest in Amazon SQS, the service must have permission to use the CMK on the caller's behalf. For more information, see [Key Management](#) in the [Amazon Simple Queue Service Developer Guide](#).

## Step Functions Action

### Step Functions Action

A `stepFunctions` action starts execution of a Step Functions state machine.

[More information \(19\)](#)

When you create a rule with a `stepFunctions` action, you must specify the following information:

`executionNamePrefix`

Optional. The name given to the state machine execution consists of this prefix followed by a UUID. Step Functions creates a unique name for each state machine execution if one is not provided.

`stateMachineName`

The name of the Step Functions state machine whose execution will be started.

`roleArn`

The ARN of the role that grants AWS IoT permission to start execution of a state machine ("Action":"states:StartExecution").

Here is an example trust policy that should be attached to the role:

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

```
        "Effect": "Allow",
        "Action": "states:StartExecution",
        "Resource": [ * ]
    }
}
```

The following JSON example shows how to create an AWS IoT rule with a `stepFunctions` action:

```
{
    "topicRulePayload": {
        "sql": "expression",
        "ruleDisabled": false,
        "description": "A step functions test rule",
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "stepFunctions": {
                    "executionNamePrefix": "myExecution",
                    "stateMachineName": "myStateMachine",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_step_functions"
                }
            }
        ]
    }
}
```

For more information, see the [AWS Step Functions Developer Guide](#).

## 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, see [Setting Up CloudWatchLogs](#).

## Error Handling (Error Action)

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 triggered, passing the SQL statement's result.

If a problem occurs when triggering an action, the rules engine triggers an error action, if one is specified for the rule. This might happen when:

- 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 receives one message that contains both errors.

The error action message looks like this:

```
{
```

```

"ruleName": "TestAction",
"topic": "testme/action",
"cloudwatchTraceId": "7e146a2c-95b5-6caf-98b9-50e3969734c7",
"clientId": "iotconsole-1511213971966-0",
"base64OriginalPayload": "ewogICJtZXNzYWdIjogIkhlbGxvIHZyb20gQVdTIElvVCBjb25zb2xIgp9",
"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: yMah1cwPhqTH267QLPhTKeVPKJB8B05ndBHzOmWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y=). Message arrived on: error/action, Action: s3, Bucket: us-east-1-s3-verify-user, Key: \"aaa\". Value of x-amz-id-2: yMah1cwPhqTH267QLPhTKeVPKJB8B05ndBHzOmWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y="
    }
]
}

```

#### ruleName

The name of the rule that triggered the error action.

#### topic

The topic on which the original message was received.

#### cloudwatchTraceld

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 about using CloudWatch to monitor the success or failure of rules, see [AWS IoT Metrics and Dimensions \(p. 199\)](#).

## Working with Topic Rule Destinations

A destination is a resource that defines where rules engine can route data. A destination can be reused across rules and might require confirmation or configuration before it can be used. Destinations make it possible for the rules engine to send data to other services that are not natively integrated with AWS IoT.

Topic rule destinations are used to verify that you own or have access to the endpoint to which you want to route data. When you create a rule action with an endpoint (for example, an `http` action) or update an existing rule action's endpoint, AWS IoT sends a confirmation message to the endpoint that contains a unique token. To verify your endpoint, you must call `ConfirmTopicRuleDestination` with the token to verify you own or have access to the endpoint. The token is valid for three days, after which you need to create a new `http` action or call the `UpdateTopicRuleDestination` API to restart the confirmation process. You can also confirm your topic rule destination by browsing to `enableUrl` or provide the token from the **Destination** page in the AWS IoT console.

When AWS IoT receives the token sent to your endpoint, the destination is confirmed. You must enable the destination before it can be used by rules engine. A destination can be in one of the following states:

### ENABLED

The destination is enabled. A destination must be in the `ENABLED` state for it to be used in a rule.  
You can only enable a destinations in `DISABLED` status.

### DISABLED

The destination has been confirmed but disabled. This is useful if you want to temporarily prevent traffic to your endpoint without having to go through the confirmation process again. You can only disable a destinations in `ENABLED` status.

### IN\_PROGRESS

Confirmation of the destination is in progress.

### ERROR

Destination confirmation timed out.

After they are confirmed and enabled, destinations can be used with any rule in your account.

## Creating a Topic Rule Destination

A destination is created when you use AWS IoT to create a rule with an `http` action. You can also create a destination using the `CreateTopicRuleDestination` API or the AWS IoT console.

When you create a destination, AWS IoT verifies the endpoint URL is valid. If the URL is valid, a confirmation message is sent to that URL. The confirmation request has the following format:

```
HTTP POST {confirmationUrl}/?confirmationToken={confirmationToken}
Headers:
x-amz-rules-engine-message-type: DestinationConfirmation
x-amz-rules-engine-destination-arn:"arn:aws:iot:us-east-1:123456789012:ruledestination/
http/7a280e37-b9c6-47a2-a751-0703693f46e4"
Content-Type: application/json
Body:
{
    "arn": "arn:aws:iot:us-east-1:123456789012:ruledestination/http/7a280e37-b9c6-47a2-
a751-0703693f46e4",
    "confirmationToken": "AYADeMXLrPrNY2wqJAKsFNn-...NBJndA",
    "enableUrl": "https://iot.us-east-1.amazonaws.com/confirmdestination/
AYADeMXLrPrNY2wqJAKsFNn-...NBJndA",
    "messageType": "DestinationConfirmation"
}
```

The fields of this message are defined as follows:

`arn`

The ARN for the topic rule destination to confirm.

`confirmationToken`

The confirmation token. The token in the example is truncated. Your token will be longer.

`enableUrl`

The URL to which you browse to confirm a topic rule destination.

`messageType`

The type of message.

## Confirming a Topic Rule Destination

You can confirm a destination by making an HTTP GET request to the `enableUrl` that is included in the confirmation message. You can call `ConfirmTopicRuleDestination` with the token from the confirmation message or you can use the AWS IoT console.

The token must be valid for the destination to be put in the `ENABLED` state. If the token has expired, you must call `UpdateTopicRuleDestination` to restart the confirmation process.

**Note**

If you confirm the topic rule destination by calling `ConfirmTopicRuleDestination`, the destination status will be `DISABLED` and you need to explicitly enable your destination by calling `UpdateTopicRuleDestination` before using it.

## Disabling a Topic Rule Destination

To disable a destination, call `UpdateTopicRuleDestination` and set the topic rule destination's status to `DISABLED`.

## Enabling a Topic Rule Destination

To enable a destination, call `UpdateTopicRuleDestination` and set the topic rule's status to `ENABLED`. You do not need to re-validate the URL.

## Sending a New Confirmation Message

To trigger a new confirmation message for a destination, call `UpdateTopicRuleDestination` and set the topic rule destination's status to `IN_PROGRESS`.

## Deleting a Topic Rule Destination

To delete a topic rule destination, call `DeleteTopicRuleDestination`.

# Reducing Messaging Costs with Basic Ingest

Basic Ingest enables you to securely send device data to the AWS services supported by [AWS IoT Rule Actions \(p. 267\)](#) without incurring [messaging costs](#). Basic Ingest optimizes data flow by removing the publish/subscribe message broker from the ingestion path, so it's more cost effective.

To use Basic Ingest, you send messages from your devices or applications with topic names that start with `$aws/rules/rule-name` as their first three levels, where `rule-name` is the name of your AWS IoT rule to trigger.

You can use an existing rule with Basic Ingest simply by adding the Basic Ingest prefix (`$aws/rules/rule-name`) to the message topic by which you normally trigger the rule. For example, if you have a rule named `BuildingManager` that is triggered by messages with topics like `Buildings/Building5/Floor2/Room201/Lights` ("sql": "SELECT \* FROM 'Buildings/#'"), you can trigger the same rule with Basic Ingest by sending a message with topic `$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights`.

Be aware that:

- Your devices and rules cannot subscribe to Basic Ingest reserved topics. For more information, see [Reserved Topics \(p. 245\)](#).
- If you need a publish/subscribe broker to distribute messages to multiple subscribers (for example, to deliver messages to other devices and the rules engine), you should continue to use the AWS IoT message broker to handle the message distribution. Just publish your messages on topics other than Basic Ingest topics.

## Using Basic Ingest

Make sure your device or application is using a [policy \(p. 139\)](#) that has publish permissions on `$aws/rules/*`. Or you can specify permission for individual rules with `$aws/rules/rule-name/*` in the policy. Otherwise, your devices and applications can continue to use their existing connections with AWS IoT Core.

When the message reaches the rules engine, there is no difference in execution or error handling between rules triggered from Basic Ingest and those triggered through message broker subscriptions.

You can, of course, create rules for use with Basic Ingest. Keep in mind the following:

- The initial prefix of a Basic Ingest topic (`$aws/rules/rule-name`) isn't available to the [topic\(Decimal\) \(p. 341\)](#) function.
- If you define a rule that is triggered only with Basic Ingest, the `FROM` clause is optional in the `sql` field of the `rule` definition. It's still required if the rule is also triggered by other messages that must be sent through the message broker (for example, because those other messages must be distributed to multiple subscribers). For more information, see [AWS IoT SQL Reference \(p. 294\)](#).
- The first three levels of the Basic Ingest topic (`$aws/rules/rule-name`) are not counted toward the eight segment length limit or toward the 256 total character limit for a topic. Otherwise, the same restrictions apply as documented in [AWS IoT Limits](#).
- If a message is received with a Basic Ingest topic that specifies an inactive rule or a rule that doesn't exist, an error log is created in an Amazon CloudWatch log to help you with debugging. For more information, see [Rules Engine Logs \(p. 220\)](#). A `RuleNotFound` metric is indicated and you can create alarms on this metric. For more information, see Rule Metrics in [AWS IoT Metrics \(p. 199\)](#).
- You can still publish with QoS 1 on Basic Ingest topics. You receive a `PUBACK` after the message is successfully delivered to the rules engine. Receiving a `PUBACK` does not mean that your rule actions were completed successfully. You can configure an error action to handle errors during action execution. See [Error Handling \(Error Action\) \(p. 289\)](#).

## 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 message payload and performs transformations.

The `SELECT` clause supports [Data Types \(p. 298\)](#), [Operators \(p. 301\)](#), [Functions \(p. 307\)](#), [Literals \(p. 342\)](#), [Case Statements \(p. 343\)](#), [JSON Extensions \(p. 343\)](#), [Substitution Templates \(p. 344\)](#), and [Nested Object Queries \(p. 345\)](#).

### FROM

The MQTT message topic filter. The rule is triggered for each message sent to an MQTT topic that matches the filter specified here. Required for rules that are triggered by messages that pass through the message broker. Optional for rules that are only triggered using the [Basic Ingest \(p. 293\)](#) feature.

### WHERE

Optional. Adds conditional logic that determines if the actions specified by a rule are carried out.

The `WHERE` clause supports [Data Types \(p. 298\)](#), [Operators \(p. 301\)](#), [Functions \(p. 307\)](#), [Literals \(p. 342\)](#), [Case Statements \(p. 343\)](#), [JSON Extensions \(p. 343\)](#), [Substitution Templates \(p. 344\)](#), and [Nested Object Queries \(p. 345\)](#).

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 color 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 \(p. 267\)](#).

## SELECT Clause

The AWS IoT SELECT clause is essentially the same as the ANSI SQL SELECT clause, with some minor differences.

The SELECT clause supports [Data Types \(p. 298\)](#), [Operators \(p. 301\)](#), [Functions \(p. 307\)](#), [Literals \(p. 342\)](#), [Case Statements \(p. 343\)](#), [JSON Extensions \(p. 343\)](#), [Substitution Templates \(p. 344\)](#), and [Nested Object Queries \(p. 345\)](#).

You can use the SELECT clause to extract information from incoming MQTT messages. You can also use `SELECT *` 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 contains 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 fahrenheit FROM 'a/b'
Outgoing payload: {"my_color":"red", "fahrenheit":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. With SQL version 2015-10-08, you can select only one item. With SQL version 2016-03-23 or later, you can also select an array to output as a top-level object.

### 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 \(p. 307\)](#)) to transform the incoming payload. You can use parentheses 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.

To use \* to refer to the message payload as raw binary data, follow these rules:

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

You can use the following `SELECT` clause with binary payloads because it doesn't refer to any JSON names.

```
SELECT * FROM 'a/b'
```

You cannot use the following `SELECT` with binary payloads because it refers to `device_type` in the `WHERE` clause.

```
SELECT * FROM 'a/b' WHERE device_type = 'thermostat'
```

You cannot use the following `SELECT` with binary payloads because it violates rule #2.

```
SELECT *, timestamp() AS timestamp FROM 'a/b'
```

You can use the following `SELECT` with binary payloads because it complies with rule #1 or rule #2.

```
SELECT * FROM 'a/b' WHERE timestamp() % 12 = 0
```

You cannot use the following AWS IoT rule with binary 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. You must enclose the topic or topic filter in single quotes (''). The rule is triggered for each message sent to an MQTT topic that matches the topic filter specified here. 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. 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'.

### # Wildcard Example:

You can use the '#' (multi-level) wildcard character to match one or more particular path elements:

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 is not triggered for the {temperature: 80} message.

### + Wildcard Example:

You can use the '+' (single-level) wildcard character to match any one particular path element:

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'.

## WHERE Clause

The WHERE clause determines if the actions specified by a rule are carried out. If the WHERE clause evaluates to true, the rule actions are performed. Otherwise, the rule actions are not performed.

The WHERE clause supports [Data Types \(p. 298\)](#), [Operators \(p. 301\)](#), [Functions \(p. 307\)](#), [Literals \(p. 342\)](#), [Case Statements \(p. 343\)](#), [JSON Extensions \(p. 343\)](#), [Substitution Templates \(p. 344\)](#), and [Nested Object Queries \(p. 345\)](#).

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 be triggered, but the actions specified by the rule would not be performed. There would be no outgoing payload.

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.

## 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.<br><b>Note</b><br>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:<br><code>"SELECT NULL AS n FROM 'a/b'"</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Undefined | <p>Not a value. This isn't explicitly representable in JSON except by omitting the value. For example, in the object <code>{ "foo": null }</code>, 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.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <code>{"foo":null, "bar":undefined}</code> </div> <p>is serialized to JSON as:</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <code>{"foo":null}</code> </div> <p>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) return Undefined.</p> |

## 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                                                                                                               |
|---------------|----------------------------------------------------------------------------------------------------------------------|
|               | \d+)?((?i)E-?\d+)?\$. "0", "-1.2", "5E-12" are all examples of strings that are converted automatically 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 tries to parse the string as a Decimal. AWS IoT attempts to parse strings matching the regular expression: ^-?\d+(\.\d+)?((?i)E-?\d+)?\$. "0", "-1.2", "5E-12" are all examples of strings that are converted automatically to Decimals. AWS IoT attempts to convert the String to a Decimal, and then truncates 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 are 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 is a comma-separated list, enclosed in square brackets. A String is quoted. A Decimal, Int, Boolean, and Null is not.                              |
| Object        | The object serialized to JSON. The resultant string is a comma-separated list of key-value pairs and begins and ends with curly braces. A String is quoted. A Decimal, Int, Boolean, and Null is not. |
| Null          | Undefined.                                                                                                                                                                                            |
| Undefined     | Undefined.                                                                                                                                                                                            |

## Operators

The following operators can be used in SELECT 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 <code>boolean</code> AND <code>boolean</code> . |
| 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 <code>boolean</code> OR <code>boolean</code> . |
| 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/<br>Decimal | String/Int/<br>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/<br>Decimal | String/Int/<br>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/<br>Decimal | String/Int/<br>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/<br>Decimal | String/Int/<br>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/<br>Decimal | String/Int/<br>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/<br>Decimal | String/Int/<br>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/<br>Decimal | String/Int/<br>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/<br>Decimal | String/Int/<br>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 2015-10-08 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 2015-10-08 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 2015-10-08 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 <code>Undefined</code> .                                                   |
| Decimal       | Decimal (with double precision), the inverse cosine of the argument. Imaginary results are returned as <code>Undefined</code> .                                                   |
| Boolean       | <code>Undefined</code> .                                                                                                                                                          |
| String        | Decimal, the inverse cosine of the argument. If the string cannot be converted, the result is <code>Undefined</code> . Imaginary results are returned as <code>Undefined</code> . |
| Array         | <code>Undefined</code> .                                                                                                                                                          |
| Object        | <code>Undefined</code> .                                                                                                                                                          |
| Null          | <code>Undefined</code> .                                                                                                                                                          |
| Undefined     | <code>Undefined</code> .                                                                                                                                                          |

## 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 2015-10-08 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 <code>Undefined</code> .                                                                           |
| Decimal       | Decimal (with double precision), the inverse sine of the argument. Imaginary results are returned as <code>Undefined</code> .                                                                           |
| Boolean       | <code>Undefined</code> .                                                                                                                                                                                |
| String        | Decimal (with double precision), the inverse sine of the argument. If the string cannot be converted, the result is <code>Undefined</code> . Imaginary results are returned as <code>Undefined</code> . |
| Array         | <code>Undefined</code> .                                                                                                                                                                                |
| Object        | <code>Undefined</code> .                                                                                                                                                                                |

| 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 2015-10-08 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 2015-10-08 and later.

Example: `atan2(1, 0) = 1.5707963267948966`

| Argument Type      | Argument Type      | Result                                                                                                                              |
|--------------------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| Int/Decimal        | Int/Decimal        | Decimal (with double precision), the angle between the positive x-axis and the specific point defined by the two arguments.         |
| Int/Decimal/String | Int/Decimal/String | Decimal, the inverse tangent of the point defined by the two arguments. If the string cannot be converted, the result is Undefined. |
| 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                                                                        |
|--------------------------|------------------------------------------------------------------------------------|
| <code>functionArn</code> | The ARN of the Lambda function to call. The Lambda function must return JSON data. |
| <code>inputJson</code>   | 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. To allow AWS IoT to invoke a Lambda function, 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](#).

Given a JSON message payload like:

```
{
    "attribute1": 21,
    "attribute2": "value"
}
```

The `aws_lambda` function can be used to call Lambda function as follows:

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function",
 {"payload":attribute1}) as output FROM 'a/b'
```

If you want to pass the full MQTT message payload, you can specify the JSON payload using `*`. For example:

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function", *) 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

The rules engine limits the execution duration of Lambda functions. Lambda function calls from rules should be completed within 2000 milliseconds.

## bitand(Int, Int)

Performs a bitwise AND on the bit representations of the two `Int`(-converted) arguments. Supported by SQL version 2015-10-08 and later.

Example: `bitand(13, 5) = 5`

| Argument Type                   | Argument Type                   | Result                                                                                                                                                                                                              |
|---------------------------------|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Int</code>                | <code>Int</code>                | <code>Int</code> , a bitwise AND of the arguments.                                                                                                                                                                  |
| <code>Int/Decimal</code>        | <code>Int/Decimal</code>        | <code>Int</code> , a bitwise AND of the arguments. If the numbers are rounded, the result is <code>Undefined</code> .                                                                                               |
| <code>Int/Decimal/String</code> | <code>Int/Decimal/String</code> | <code>Int</code> , a bitwise AND of the arguments. If the numbers are converted to decimal, the result is the nearest <code>Int</code> . If the result is <code>NaN</code> , the result is <code>Undefined</code> . |
| Other Value                     | Other Value                     | <code>Undefined</code> .                                                                                                                                                                                            |

## bitor(Int, Int)

Performs a bitwise OR of the bit representations of the two arguments. Supported by SQL version 2015-10-08 and later.

Example: `bitor(8, 5) = 13`

| Argument Type      | Argument Type      | Result                                                                                                                                                                                                                                                                                                         |
|--------------------|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Int                | Int                | Int, the bitwise OR of the two Int(-converted) arguments.                                                                                                                                                                                                                                                      |
| Int/Decimal        | Int/Decimal        | Int, the bitwise OR of the two Int(-converted) arguments. If either number is a decimal, it is converted to an integer. If the conversion fails, the result is Undefined.                                                                                                                                      |
| Int/Decimal/String | Int/Decimal/String | Int, the bitwise OR of the two Int(-converted) arguments. If either argument is a string, it is converted to a decimal. If the conversion fails, the result is Undefined. If the conversion succeeds, the result is an Int. If the conversion succeeds and the result is a decimal, it is converted to an Int. |
| 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 2015-10-08 and later.

Example: `bitxor(13, 5) = 8`

| Argument Type      | Argument Type      | Result                                                                                                                                                                                                                                                                                                        |
|--------------------|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Int                | Int                | Int, a bitwise XOR of the two Int(-converted) arguments.                                                                                                                                                                                                                                                      |
| Int/Decimal        | Int/Decimal        | Int, a bitwise XOR of the two Int(-converted) arguments. If either number is a decimal, it is converted to an integer. If the conversion fails, the result is Undefined.                                                                                                                                      |
| Int/Decimal/String | Int/Decimal/String | Int, a bitwise XOR of the two Int(-converted) arguments. If either argument is a string, it is converted to a decimal. If the conversion fails, the result is Undefined. If the conversion succeeds, the result is an Int. If the conversion succeeds and the result is a decimal, it is converted to an Int. |
| Other Value        | Other Value        | Undefined.                                                                                                                                                                                                                                                                                                    |

## bitnot(Int)

Performs a bitwise NOT on the bit representations of the `Int`(-converted) argument. Supported by SQL version 2015-10-08 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 or from Booleans. If AWS IoT cannot determine how to cast one type to another, the result is Undefined. Supported by SQL version 2015-10-08 and later. Format: `cast(value as type)`.

Example:

```
cast(true as Decimal) = 1.0
```

The following keywords might appear after "as" when calling cast:

**For SQL version 2015-10-08 and 2016-03-23**

| 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.     |
| Integer  | Casts value to Int.     |

**Additionally, for SQL version 2016-03-23**

| Keyword | Result                  |
|---------|-------------------------|
| Decimal | Casts value to Decimal. |
| Bool    | Casts value to Boolean. |
| Boolean | Casts value to Boolean. |

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.         |

| Argument Type | Result                                                                                                                                                                                                                    |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| String        | Tries to parse the string as a Decimal. AWS IoT attempts to parse strings matching the regex: ^-?\d+(.\d+)?((?i)E-?\d+)?\$."0", "-1.2", "5E-12" are all examples of strings that are converted automatically to decimals. |
| 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.                                                                                                                                                                                                                                                                                         |
| String        | Tries to parse the string as a Decimal. AWS IoT attempts to parse strings matching the regex: ^-?\d+(.\d+)?((?i)E-?\d+)?\$."0", "-1.2", "5E-12" are all examples of strings that are converted automatically to decimals. AWS IoT attempts 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.                                                                             |

| Argument Type | Result     |
|---------------|------------|
| 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 is a comma-separated list enclosed in square brackets. String is quoted. Decimal, Int, and Boolean are not.                                   |
| Object        | The object serialized to JSON. The JSON string is a comma-separated list of key-value pairs and begins and ends with curly braces. String is quoted. Decimal, Int, Boolean, and Null are not. |
| Null          | Undefined.                                                                                                                                                                                    |
| Undefined     | Undefined.                                                                                                                                                                                    |

## ceil(Decimal)

Rounds the given Decimal up to the nearest Int. Supported by SQL version 2015-10-08 and later.

Examples:

`ceil(1.2) = 2`

`ceil(-1.2) = -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 2015-10-08 and later.

Examples:

```
chr(65) = "A".
```

```
chr(49) = "1".
```

| Argument Type        | Result                                                                                                                                                                                                                           |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Int</code>     | The character corresponding to the specified ASCII value. If the argument is not a valid ASCII value, the result is <code>Undefined</code> .                                                                                     |
| <code>Decimal</code> | The character corresponding to the specified ASCII value. The <code>Decimal</code> argument is rounded down to the nearest <code>Int</code> . If the argument is not a valid ASCII value, the result is <code>Undefined</code> . |
| <code>Boolean</code> | <code>Undefined</code> .                                                                                                                                                                                                         |
| <code>String</code>  | If the <code>String</code> can be converted to a <code>Decimal</code> , it is rounded down to the nearest <code>Int</code> . If the argument is not a valid ASCII value, the result is <code>Undefined</code> .                  |
| <code>Array</code>   | <code>Undefined</code> .                                                                                                                                                                                                         |
| <code>Object</code>  | <code>Undefined</code> .                                                                                                                                                                                                         |
| <code>Null</code>    | <code>Undefined</code> .                                                                                                                                                                                                         |
| Other Value          | <code>Undefined</code> .                                                                                                                                                                                                         |

## 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 2015-10-08 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 2015-10-08 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 <b>Array</b> , the result is a single array containing all of the arguments. If no arguments are arrays, and at least one argument is a <b>String</b> , the result is the concatenation of the <b>String</b> representations of all the arguments. Arguments are converted to strings using the standard conversions listed above.<br>. |

## cos(Decimal)

Returns the cosine of a number in radians. **Decimal** arguments are rounded to double precision before function application. Supported by SQL version 2015-10-08 and later.

Example:

```
cos(0) = 1.
```

| Argument Type    | Result                                                                                                                                                                                                            |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Int</b>       | <b>Decimal</b> (with double precision), the cosine of the argument. Imaginary results are returned as <b>Undefined</b> .                                                                                          |
| <b>Decimal</b>   | <b>Decimal</b> (with double precision), the cosine of the argument. Imaginary results are returned as <b>Undefined</b> .                                                                                          |
| <b>Boolean</b>   | <b>Undefined</b> .                                                                                                                                                                                                |
| <b>String</b>    | <b>Decimal</b> (with double precision), the cosine of the argument. If the string cannot be converted to a <b>Decimal</b> , the result is <b>Undefined</b> . Imaginary results are returned as <b>Undefined</b> . |
| <b>Array</b>     | <b>Undefined</b> .                                                                                                                                                                                                |
| <b>Object</b>    | <b>Undefined</b> .                                                                                                                                                                                                |
| <b>Null</b>      | <b>Undefined</b> .                                                                                                                                                                                                |
| <b>Undefined</b> | <b>Undefined</b> .                                                                                                                                                                                                |

## 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 2015-10-08 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 <code>Undefined</code> .                                                                                        |
| Decimal       | Decimal (with double precision), the hyperbolic cosine of the argument. Imaginary results are returned as <code>Undefined</code> .                                                                                        |
| Boolean       | <code>Undefined</code> .                                                                                                                                                                                                  |
| String        | Decimal (with double precision), the hyperbolic cosine of the argument. If the string cannot be converted to a Decimal, the result is <code>Undefined</code> . Imaginary results are returned as <code>Undefined</code> . |
| Array         | <code>Undefined</code> .                                                                                                                                                                                                  |
| Object        | <code>Undefined</code> .                                                                                                                                                                                                  |
| Null          | <code>Undefined</code> .                                                                                                                                                                                                  |
| Undefined     | <code>Undefined</code> .                                                                                                                                                                                                  |

## 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 \(p. 294\)](#). 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 is 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 2015-10-08 and later.

Example: `endswith("cat", "at") = true.`

| argument Type 1 | argument Type 2 | Result                                                                                                                                                                                                                                         |
|-----------------|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| String          | String          | True if the first argument is a String and the second argument is a standard conversion of the first. Otherwise, false.                                                                                                                        |
| Other Value     | Other Value     | Both arguments are converted to strings using standard conversion rules. If either argument ends in the second argument, the result is true. Otherwise, the result is false. If either argument is null or undefined, the result is undefined. |

## exp(Decimal)

Returns e raised to the Decimal argument. Decimal arguments are rounded to double precision before function application. Supported by SQL version 2015-10-08 and later.

Example: `exp(1) = e.`

| Argument Type | Result                                                                                                                            |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Int           | Decimal (with double precision), $e^{\text{argument}}$ .                                                                          |
| Decimal       | Decimal (with double precision), $e^{\text{argument}}$ .                                                                          |
| String        | Decimal (with double precision), $e^{\text{argument}}$ . If the String cannot be converted to a Decimal, the result is Undefined. |
| Other Value   | Undefined.                                                                                                                        |

## floor(Decimal)

Rounds the given Decimal down to the nearest Int. Supported by SQL version 2015-10-08 and later.

Examples:

`floor(1.2) = 1`

`floor(-1.2) = -2`

| Argument Type | Result                                                                                                                                                |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Int           | Int, the argument value.                                                                                                                              |
| Decimal       | Int, the Decimal value rounded down to the nearest Int.                                                                                               |
| String        | Int. The string is converted to Decimal and rounded down to the nearest Int. 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 is applied to the first argument. Conversion applies as documented in the table to the second argument. Supported by SQL version 2015-10-08 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-based index specified by the second argument. If conversion is unsuccessful or the index is outside the bounds of array.length), the result is undefined.       |
| String          | Any Type (converted to Int)       | The character at the 0-based index specified by the second argument. If conversion is unsuccessful or the index is outside the bounds of string.length), the result is undefined. |
| Object          | String (no conversion is applied) | The value stored in the object corresponding to the key argument.                                                                                                                 |
| Other Value     | Any Value                         | Undefined.                                                                                                                                                                        |

## get\_dynamodb(tableName, partitionKeyName, partitionKeyValue, sortKeyName, sortKeyValue, roleArn)

Retrieves data from a DynamoDB table. `get_dynamodb()` allows you to query a DynamoDB table while a rule is evaluated. You can filter or augment message payloads using data retrieved from DynamoDB. Supported by SQL version 2016-03-23 and later.

`get_dynamodb()` takes the following parameters:

`tableName`

The name of the DynamoDB table to query.

`partitionKeyName`

The name of the partition key. For more information, see [DynamoDB Keys](#).

`partitionKeyValue`

The value of the partition key used to identify a record. For more information, see [DynamoDB Keys](#).

`sortKeyName`

Optional. The name of the sort key. This parameter is required only if the DynamoDB table queried uses a composite key. For more information, see [DynamoDB Keys](#).

#### sortKeyValue

Optional. The value of the sort key. This parameter is required only if the DynamoDB table queried uses a composite key. For more information, see [DynamoDB Keys](#).

#### roleArn

The ARN of an IAM role that grants access to the DynamoDB table. The rules engine assumes this role to access the DynamoDB table on your behalf. Avoid using an overly permissive role. Grant the role only those permissions required by the rule. The following is an example policy that grants access to one DynamoDB table.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "dynamodb:GetItem",  
            "Resource": "arn:aws:dynamodb:aws-region:account-id:table/table-name"  
        }  
    ]  
}
```

As an example of how you can use `get_dynamodb()`, say you have a DynamoDB table that contains device ID and location information for all of your devices connected to AWS IoT. The following SELECT statement uses the `get_dynamodb()` function to retrieve the location for the specified device ID:

```
SELECT *, get_dynamodb("InServiceDevices", "deviceId", id,  
"arn:aws:iam::12345678910:role/getdynamo").location AS location FROM 'some/  
topic'
```

#### Note

- You can call `get_dynamodb()` a maximum of one time per SQL statement. Calling `get_dynamodb()` multiple times in a single SQL statement causes the rule to terminate without invoking any actions.
- If `get_dynamodb()` returns more than 8 KB of data, the rule's action may not be invoked.

## 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 2015-10-08 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 2015-10-08 and later.

Examples:

```
indexof("abcd", "bc") = 1
```

## isNull()

Returns true if the argument is the Null value. Supported by SQL version 2015-10-08 and later.

Examples:

```
isNull(5) = false.
```

```
isNull(NULL) = true.
```

| Argument Type | Result |
|---------------|--------|
| Int           | false  |
| Decimal       | false  |
| Boolean       | false  |
| String        | false  |
| Array         | false  |
| Object        | false  |
| Null          | true   |

| Argument Type | Result |
|---------------|--------|
| Undefined     | false  |

## isUndefined()

Returns true if the argument is Undefined. Supported by SQL version 2016-03-23 and later.

Examples:

```
isUndefined(5) = false.  
isUndefined(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 2015-10-08 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 2015-10-08 and later.

Example:  $\ln(e) = 1$ .

| Argument Type | Result                                                            |
|---------------|-------------------------------------------------------------------|
| Int           | Decimal (with double precision), the natural log of the argument. |

| Argument Type | Result                                                                                                                                     |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| 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 2015-10-08 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 2015-10-08 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 `Int` argument. The `Int` argument must be between 0 and 1000. If the provided value is outside of this valid range, the argument is set to the nearest valid value (0 or 1000). Supported by SQL version 2015-10-08 and later.

Examples:

`lpad("hello", 2) = " hello".`

`lpad(1, 3) = " 1"`

| argument Type 1 | argument Type 2    | Result                                                                                                                                            |
|-----------------|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| String          | Int                | String, the provided String with a number of spaces.                                                                                              |
| String          | Decimal            | The Decimal argument is converted to Int and the String is padded with the specified number of spaces.                                            |
| String          | String             | The second argument is rounded down to the nearest integer and the String is padded with the specified number of spaces. The result is Undefined. |
| Other Value     | Int/Decimal/String | The first value is converted to standard conversions and applied on that String. The result is Undefined.                                         |
| Any Value       | Other Value        | Undefined.                                                                                                                                        |

## ltrim(String)

Removes all leading white space (tabs and spaces) from the provided `String`. Supported by SQL version 2015-10-08 and later.

Example:

`Ltrim(" h i ") = "hi".`

| Argument Type | Result                                                                                             |
|---------------|----------------------------------------------------------------------------------------------------|
| Int           | The String representation of the Int with all leading white space removed.                         |
| Decimal       | The String representation of the Decimal with all leading white space removed.                     |
| Boolean       | The String representation of the boolean ("true" or "false") with all leading white space removed. |
| String        | The argument with all leading white space removed.                                                 |

| Argument Type | Result                                                                                                          |
|---------------|-----------------------------------------------------------------------------------------------------------------|
| Array         | The String representation of the Array (using standard conversion rules) with all leading white space removed.  |
| Object        | The String representation of the Object (using standard conversion rules) with all leading white space 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 2015-10-08 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 is 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. Equivalent to [remainder\(Decimal, Decimal\) \(p. 332\)](#). You can also use "%" as an infix operator for the same modulo functionality. Supported by SQL version 2015-10-08 and later.

Example: `mod(8, 3) = 2`.

| Left Operand       | Right Operand      | Output                                                                                              |
|--------------------|--------------------|-----------------------------------------------------------------------------------------------------|
| Int                | Int                | Int, the first argument                                                                             |
| Int/Decimal        | Int/Decimal        | Decimal, the first argument                                                                         |
| String/Int/Decimal | String/Int/Decimal | If all strings convert to Int, the first argument modulo the second argument. Otherwise, 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 2015-10-08 and later.

Example: `Nanvl(8, 3) = 8`.

| argument Type 1   | argument Type 2 | Output              |
|-------------------|-----------------|---------------------|
| Undefined         | Any Value       | The second argument |
| Null              | Any Value       | The second argument |
| Decimal (NaN)     | Any Value       | The second argument |
| 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 2015-10-08 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 2015-10-08 and later.

Examples:

```
numbytes("hi") = 2
numbytes("€") = 3
```

## principal()

Returns the principal the device uses for authentication. Which principal is returned depends on how the triggering message was published. The following table describes the principal returned for each publishing method and protocol.

| How message is published    | Protocol            | Credential Type          |
|-----------------------------|---------------------|--------------------------|
| MQTT client                 | MQTT                | X.509 device certificate |
| AWS IoT console MQTT client | MQTT                | IAM user or role         |
| AWS CLI                     | HTTP                | IAM user or role         |
| AWS IoT Device SDK          | MQTT                | X.509 device certificate |
| AWS IoT Device SDK          | MQTT over WebSocket | IAM user or role         |

The following examples show the different types of values returned by `principal`:

- X.509 certificate thumbprint:  
`ba67293af50bf2506f5f93469686da660c7c844e7b3950bfb16813e0d31e9373`
- IAM role ID and session name: `ABCD1EFG3HIJK2LMNOP5:my-session-name`
- returns a user ID: `ABCD1EFG3HIJK2LMNOP5`

## 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 that 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\(\) \(p. 340\)](#).

timezone

(String) Optional. The time zone of the formatted date/time. The default is "UTC". The function supports [Joda-Time time zones](#).

Examples:

When this message is published to the topic 'A/B', the payload `{"ts": "1970.01.01 AD at 21:46:40 CST"}` is sent to the S3 bucket:

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT parse_time(\"yyyy.MM.dd G 'at' HH:mm:ss z\", 100000000, \"America/Belize\") as ts FROM 'A/B'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "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) is sent to the S3 bucket:

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT parse_time(\"yyyy.MM.dd G 'at' HH:mm:ss z\", timestamp() ) as ts FROM 'A/B'",
        "awsIotSqlVersion": "2016-03-23",
        "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 is sent to the S3 bucket with key = "2017".

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT * FROM 'A/B'",
        "awsIotSqlVersion": "2016-03-23",
        "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 2015-10-08 and later. Supported by SQL version 2015-10-08 and later.

Example: `power(2, 5) = 32.0.`

| argument Type 1    | argument Type 2    | Output                                                                                                                         |
|--------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------|
| Int/Decimal        | Int/Decimal        | A Decimal (with double precision) raised to the second argument.                                                               |
| Int/Decimal/String | Int/Decimal/String | A Decimal (with double precision) raised to the second argument. All inputs are converted to decimal and converted to Decimal. |
| Other Value        | Other Value        | Undefined.                                                                                                                     |

## rand()

Returns a pseudorandom, uniformly distributed double between 0.0 and 1.0. Supported by SQL version 2015-10-08 and later.

Example:

`rand() = 0.8231909191640703`

## regexp\_matches(String, String)

Returns true if the string (first argument) contains a match for the regular expression (second argument).

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.                                                   |

| Argument Type | Result                                                                     |
|---------------|----------------------------------------------------------------------------|
| 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 might not be a valid regular expression. If the (converted) argument is not valid regex, 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 2015-10-08 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 using the standard conversion rules. Depending on the type, the resultant string might 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 are converted to `String` 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 second parameter (regex) in the first parameter. Reference capture groups with "\$". Supported by SQL version 2015-10-08 and later.

**Example:**

```
regexp_substr("hihihello", "hi") = "hi"
regexp_substr("hihihello", "(hi)*") = "hihi"
```

### First argument:

| Argument Type          | Result                                                                                               |
|------------------------|------------------------------------------------------------------------------------------------------|
| <code>Int</code>       | The <code>String</code> representation of the <code>Int</code> .                                     |
| <code>Decimal</code>   | The <code>String</code> representation of the <code>Decimal</code> .                                 |
| <code>Boolean</code>   | The <code>String</code> representation of the boolean ("true" or "false").                           |
| <code>String</code>    | The <code>String</code> argument.                                                                    |
| <code>Array</code>     | The <code>String</code> representation of the <code>Array</code> (using standard conversion rules).  |
| <code>Object</code>    | The <code>String</code> representation of the <code>Object</code> (using standard conversion rules). |
| <code>Null</code>      | <code>Undefined</code> .                                                                             |
| <code>Undefined</code> | <code>Undefined</code> .                                                                             |

### 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 might 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 are converted to `String` using the standard conversion rules. If the argument is not a valid regex replacement string, the result is `Undefined`.

## remainder(Decimal, Decimal)

Returns the remainder of the division of the first argument by the second argument. Equivalent to [mod\(Decimal, Decimal\) \(p. 327\)](#). You can also use "%" as an infix operator for the same modulo functionality. Supported by SQL version 2015-10-08 and later.

**Example:** `remainder(8, 3) = 2`.

| Left Operand       | Right Operand      | Output                                                                                                                                    |
|--------------------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| Int                | Int                | Int, the first argument.                                                                                                                  |
| Int/Decimal        | Int/Decimal        | Decimal, the first argument.                                                                                                              |
| String/Int/Decimal | String/Int/Decimal | If all strings convert to Int, the first argument modulo the second argument. If any string converts to Decimal, the result is Undefined. |
| Other Value        | Other Value        | Undefined.                                                                                                                                |

## replace(String, String, String)

Replaces all occurrences of the second argument in the first argument with the third argument.  
Supported by SQL version 2015-10-08 and later.

Example:

```
replace("abcd", "bc", "x") = "axd".
replace("abcdabcd", "b", "x") = "afcdafcd".
```

### All arguments

| 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.                                                                 |

## 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 is set to the nearest valid value (0 or 1000). Supported by SQL version 2015-10-08 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 of spaces equal to the provided Int.                                                             |
| String          | Decimal         | The Decimal argument is 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. |
| String          | String          | The second argument is converted to a Decimal, which is rounded down to the nearest                                                                   |

| argument Type 1 | argument Type 2    | Result                                                                                                                                                                        |
|-----------------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 |                    | Int.<br>The String is padded on the right side with a number of spaces equal to the Int value.                                                                                |
| Other Value     | Int/Decimal/String | The first value is converted to a String using the standard conversions, and the rpad function is 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 2015-10-08 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. |
| Other Value   | Undefined.                                                                                                           |

## rtrim(String)

Removes all trailing white space (tabs and spaces) from the provided String. Supported by SQL version 2015-10-08 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 2015-10-08 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 2015-10-08 and later. |
| 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 2015-10-08 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 2015-10-08 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.                                                                                                                                     |
| 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 are set to zero. Indices that are greater than the String length are 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 are 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 2015-10-08 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) = "true".
substring(false, -2.411E247) = "false".
substring("012345", 1, 3) = "12".
substring("012345", -50, 50) = "012345".
substring("012345", 3, 1) = "".
```

## sql\_version()

Returns the SQL version specified in this rule. Supported by SQL version 2015-10-08 and later.

Example:

```
sql_version() = "2016-03-23"
```

## sqrt(Decimal)

Returns the square root of a number. Decimal arguments are rounded to double precision before function application. Supported by SQL version 2015-10-08 and later.

Example: `sqrt(9) = 3.0.`

| Argument Type | Result                                                                                                    |
|---------------|-----------------------------------------------------------------------------------------------------------|
| Int           | The square root of the argument.                                                                          |
| Decimal       | The square root of the argument.                                                                          |
| 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 2015-10-08 and later.

Example:

```
startswith("ranger", "ran") = true
```

| argument Type 1 | argument Type 2 | Result                                                                                          |
|-----------------|-----------------|-------------------------------------------------------------------------------------------------|
| String          | String          | Whether the first str                                                                           |
| Other Value     | Other Value     | Both arguments are c<br>standard conversion n<br>starts with the second<br>or Undefined, the re |

## tan(Decimal)

Returns the tangent of a number in radians. Decimal values are rounded to double precision before function application. Supported by SQL version 2015-10-08 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.                                                                                                                             |
| 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 2015-10-08 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 2015-10-08 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 topic segment, with `1` designating the first segment. For the topic `foo/bar/baz`, `topic(1)` returns `foo`, `topic(2)` returns `bar`, and so on. Supported by SQL version 2015-10-08 and later.

Examples:

```
topic() = "things/myThings/thingOne"
```

```
topic(1) = "things"
```

When [Basic Ingest \(p. 293\)](#) is used, the initial prefix of the topic (`$aws/rules/rule-name`) is not available to the `topic()` function. For example, given the topic:

```
$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights
topic() = "Buildings/Building5/Floor2/Room201/Lights"
topic(3) = "Floor2"
```

## traceid()

Returns the trace ID (UUID) of the MQTT message, or `Undefined` if the message wasn't sent over MQTT. Supported by SQL version 2015-10-08 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 is set to zero. If the second argument is greater than 34, it is set to 34. Trailing zeroes are stripped from the result. Supported by SQL version 2015-10-08 and later.

Examples:

```
trunc(2.3, 0) = 2.
```

```
trunc(2.3123, 2) = 2.31.
```

```
trunc(2.888, 2) = 2.88.
```

```
trunc(2.00, 5) = 2.
```

| argument Type 1                 | argument Type 2          | Result                                                                                                                                                                                                                                                                                                                                                                |
|---------------------------------|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Int</code>                | <code>Int</code>         | The source value.                                                                                                                                                                                                                                                                                                                                                     |
| <code>Int/Decimal</code>        | <code>Int/Decimal</code> | The first argument is converted to a <code>Decimal</code> . If the second argument is less than zero, it is set to zero. If the second argument is greater than 34, it is set to 34. The result is truncated to the specified number of decimal places. Trailing zeroes are stripped from the result.                                                                 |
| <code>Int/Decimal/String</code> | <code>Int/Decimal</code> | The first argument is converted to a <code>Decimal</code> . If the second argument is less than zero, it is set to zero. If the second argument is greater than 34, it is set to 34. The result is truncated to the specified number of decimal places. Trailing zeroes are stripped from the result. If the conversion fails, the result is <code>Undefined</code> . |

| argument Type 1 | argument Type 2 | Result     |
|-----------------|-----------------|------------|
| Other Value     |                 | Undefined. |

## trim(String)

Removes all leading and trailing white space from the provided `String`. Supported by SQL version 2015-10-08 and later.

Example:

```
Trim(" hi ") = "hi"
```

| Argument Type          | Result                                                                                                                                    |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Int</code>       | The <code>String</code> representation of the <code>Int</code> with all leading and trailing white space removed.                         |
| <code>Decimal</code>   | The <code>String</code> representation of the <code>Decimal</code> with all leading and trailing white space removed.                     |
| <code>Boolean</code>   | The <code>String</code> representation of the <code>Boolean</code> ("true" or "false") with all leading and trailing white space removed. |
| <code>String</code>    | The <code>String</code> with all leading and trailing white space removed.                                                                |
| <code>Array</code>     | The <code>String</code> representation of the <code>Array</code> using standard conversion rules.                                         |
| <code>Object</code>    | The <code>String</code> representation of the <code>Object</code> using standard conversion rules.                                        |
| <code>Null</code>      | <code>Undefined</code> .                                                                                                                  |
| <code>Undefined</code> | <code>Undefined</code> .                                                                                                                  |

## 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 2015-10-08 and later.

Examples:

```
upper("hello") = "HELLO"
upper(["hello"]) = "[\"HELLO\"]"
```

## 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 available only when using SQL version 2016-03-23 or later.

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: {"lat\_long": {"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 r[e] 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 might 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 a SELECT or WHERE clause. This includes functions, operators, and information present in the original message payload.

#### Important

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

For more information about supported expressions, see [AWS IoT SQL Reference \(p. 294\)](#).

Substitution templates appear in the action parameters within a rule:

```
{
    "sql": "SELECT *, timestamp() AS timestamp 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 published to my/iot/topic:

```
{
  "deviceid": "iot123",
  "temp": 54.98,
  "humidity": 32.43,
  "coords": {
    "latitude": 47.615694,
    "longitude": -122.3359976
  }
}
```

Then this rule publishes the following JSON to my/iot/topic/republish, which AWS IoT substitutes from \${topic()}/republish:

```
{
  "deviceid": "iot123",
  "temp": 54.98,
  "humidity": 32.43,
  "coords": {
    "latitude": 47.615694,
    "longitude": -122.3359976
  },
  "timestamp": 1579637878451
}
```

## Nested Object Queries

You can use nested SELECT clauses to query for attributes within arrays and inner JSON objects.  
Supported by SQL version 2016-03-23 and later.

Consider 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 }
  ]
}
```

### Example

You can convert values to a new array with the following rule.

```
SELECT (SELECT VALUE n FROM e) as sensors FROM 'my/topic'
```

The rule generates the following output.

```
{
  "sensors": [
    "temperature",
```

```
        "light",
        "acidity"
    ]
}
```

### Example

Using the same MQTT message, you can also query a specific value within a nested object with 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
        }
    ]
}
```

### Example

You can also flatten the output with a more complicated rule.

```
SELECT get((SELECT v FROM e WHERE n = 'temperature'), 0).v as temperature FROM 'topic'
```

The rule generates the following output.

```
{
    "temperature": 22.5
}
```

## 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 \(p. 262\)](#). The `awsIotSqlVersion` property lets you specify which version of the AWS IoT SQL rules engine that you want to use. When a new version is deployed, you can continue to use an earlier 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 you 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"
            }
        }
    ]
}
```

```
    }]
}
```

AWS IoT currently supports the following SQL versions:

- 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. If you use this version, it 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.
- Intra-object query support. For more information, see [Nested Object Queries \(p. 345\)](#).
- Support to output an array as a top-level object.
- Addition of the `encode(value, encodingScheme)` function, which can be applied on JSON and non-JSON format data. For more information, see the [encode function \(p. 318\)](#).

### 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]
```

# Device Shadow Service for AWS IoT

A device's *shadow* is a JSON document that is used to store and retrieve current state information for a device. The Device Shadow service maintains a shadow for each device you connect to AWS IoT. You can use the shadow to get and set the state of a device over MQTT or HTTP, regardless of whether the device is connected to the Internet. Each device's shadow is uniquely identified by the name of the corresponding thing.

## Contents

- [Device Shadow Service Data Flow \(p. 348\)](#)
- [Device Shadow Service Documents \(p. 355\)](#)
- [Using Shadows \(p. 359\)](#)
- [Device Shadow RESTful API \(p. 367\)](#)
- [Shadow MQTT Topics \(p. 370\)](#)
- [Shadow Document Syntax \(p. 377\)](#)
- [Shadow Error Messages \(p. 379\)](#)

## Device Shadow Service Data Flow

The Device Shadow service acts as an intermediary, allowing devices and applications to retrieve and update a device's shadow.

To illustrate how devices and applications communicate with the Device Shadow 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 Device Shadow service.

The Device Shadow 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 Device Shadow service sends messages to this topic when an update is successfully made to the device's shadow.

\$aws/things/myLightBulb/shadow/update/rejected

The Device Shadow service sends messages to this topic when an update to the device's shadow is rejected.

\$aws/things/myLightBulb/shadow/update/delta

The Device Shadow service sends messages to this topic when a difference is detected between the reported and desired sections of the device's shadow. For more information, see [/update/delta \(p. 373\)](#).

\$aws/things/myLightBulb/shadow/get/accepted

The Device Shadow service sends messages to this topic when a request for the device's shadow is made successfully.

\$aws/things/myLightBulb/shadow/get/rejected

The Device Shadow service sends messages to this topic when a request for the device's shadow is rejected.

\$aws/things/myLightBulb/shadow/delete/accepted

The Device Shadow service sends messages to this topic when the device's shadow is deleted.

\$aws/things/myLightBulb/shadow/delete/rejected

The Device Shadow service sends messages to this topic when a request to delete the device's shadow is rejected.

\$aws/things/myLightBulb/shadow/update/documents

The Device Shadow service publishes a state document to this topic whenever an update to the device's shadow is successfully performed.

To learn more about all of the MQTT topics used by the Device Shadow service, see [Shadow MQTT Topics \(p. 370\)](#).

**Note**

We recommend that you subscribe to the `.../rejected` topics to see any errors sent by the Device Shadow service.

When the light bulb comes online, it sends its current state to the Device Shadow service by sending an MQTT message to the `$aws/things/myLightBulb/shadow/update` topic.

**Note**

Device Shadows are created the first time an attempt is made to update it. The Device Shadow service will detect that a shadow doesn't exist and will create one. If the shadow exists, it will be updated.

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 Device Shadow 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 Device Shadow service received the UPDATE request and updated the device's shadow. If the shadow doesn't exist, it is created. Otherwise, the 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 Device Shadow 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 device's shadow is successfully performed. For more information of the contents of messages published to this topic, see [Shadow MQTT Topics \(p. 370\)](#).

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 Device Shadow service responds by publishing the requested 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 Device Shadow 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 Device Shadow service publishes a message to this topic when it accepts a shadow update and the resulting shadow contains different values for desired and reported states.

The Device 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 Device Shadow 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 Device Shadow 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 shadow.

The Device Shadow service returns the shadow document:

```
{  
    "state": {  
        "reported": {  
            "color": "green"  
        }  
    },  
    "metadata": {  
        "reported": {  
            "color": {  
                "timestamp": 1469564801  
            }  
        }  
    },  
    "version": 3,  
    "timestamp": 1469564864  
}
```

To delete the device's 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 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 Device Shadow service, but are still processed by subscribed clients and by the AWS IoT rules engine. If you want the Device Shadow 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 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.

## Device Shadow Service Documents

The Device Shadow service respects all rules of the JSON specification. Values, objects, and arrays are stored in the device's shadow document.

## Contents

- [Document Properties \(p. 356\)](#)
- [Versioning of a Device Shadow \(p. 356\)](#)
- [Client Token \(p. 357\)](#)
- [Example Document \(p. 357\)](#)
- [Empty Sections \(p. 357\)](#)
- [Arrays \(p. 358\)](#)

# Document Properties

A device's 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.

### Note

Metadata do not contribute to the document size for service limits or pricing. For more information, see [AWS IoT Service Limits](#).

`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 [Shadow Document Syntax \(p. 377\)](#).

## Versioning of a Device Shadow

The Device Shadow service supports versioning on every update message (both request and response), which means that with every update of a device's 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 device's 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.

### Note

The client token can be no longer than 64 bytes. A client token that is longer than 64 bytes will cause a 400 (Bad Request) response and an *Invalid clientToken* error message.

## Example Document

Here is an example 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 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 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

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 Shadows

AWS IoT provides three methods for working with a device's shadow:

### UPDATE

Creates a device's shadow if it doesn't exist, or updates the contents of a device's 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 device's 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 device's shadow, including all of its content. This removes the JSON document from the data store. You can't restore a device's shadow you deleted, but you can create a new 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 [Shadow MQTT Topics \(p. 370\)](#) and [Device Shadow RESTful API \(p. 367\)](#).

## Updating a Shadow

You can update a device's shadow by using the [UpdateThingShadow \(p. 368\)](#) RESTful API or by publishing to the [/update \(p. 370\)](#) 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 shadow JSON document.

Final state:

```
{
    "state": {
        "reported" : {
            "color" : { "r" : 10, "g" : 255, "b": 0 },
            "engine" : "ON"
        }
    }
}
```

## Retrieving a Shadow Document

You can retrieve a device's shadow by using the [GetThingShadow \(p. 368\)](#) RESTful API or by subscribing and publishing to the [/get \(p. 374\)](#) 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 device's 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 device's shadow by publishing to the [/update \(p. 370\)](#) 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 device's shadow by setting its state to `null`. For example, sending the following message deletes all of the state data, but the device's shadow remains.

```
{
  "state": null
}
```

The device's shadow still exists even if its state is `null`. The version of the shadow is incremented when the next update occurs.

## Deleting a Shadow

You can delete a device's shadow document by using the [DeleteThingShadow \(p. 369\)](#) RESTful API or by publishing to the [/delete \(p. 375\)](#) topic.

**Note**

Deleting a device's shadow does not delete the thing. Deleting a thing does not delete the corresponding device's 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 Device Shadow 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 device's 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 device's shadow document. Upon receiving this message, the device should decide whether to make the requested change. If the device's state is changed, it should publish 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. A device reports its 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 device's 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 Shadow Messages

To reduce the size of shadow messages sent to your device, define a rule that selects only the fields your device needs then 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(3)}/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 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(3) evaluates to the thing name in the original topic. For more information about creating rules, see [Rules](#).

## Device Shadow RESTful API

A 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
```

The shadow RESTful API follows the same HTTPS protocols/port mappings as described in [AWS IoT Protocols](#).

#### API Actions

- [GetThingShadow \(p. 368\)](#)
- [UpdateThingShadow \(p. 368\)](#)
- [DeleteThingShadow \(p. 369\)](#)

#### Note

When using these API, make sure you use the 8443 port as described in [Protocols, Port Mappings, and Authentication \(p. 253\)](#).

## GetThingShadow

Gets the 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 \(p. 377\)](#).

#### Authorization

Retrieving a shadow requires a policy that allows the caller to perform the `iot:GetThingShadow` action. The Device Shadow 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 device's shadow:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:GetThingShadow",
            "Resource": ["arn:aws:iot:region:account:thing/thing"]
        }
    ]
}
```

## UpdateThingShadow

Updates the 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 device's 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 \(p. 377\)](#).

### 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 \(p. 377\)](#).

### Authorization

Updating a shadow requires a policy that allows the caller to perform the `iot:UpdateThingShadow` action. The Device Shadow 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 device's shadow:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:UpdateThingShadow",
            "Resource": ["arn:aws:iot:region:account:thing/thing"]
        }
    ]
}
```

## DeleteThingShadow

Deletes the 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 device's shadow requires a policy that allows the caller to perform the `iot:DeleteThingShadow` action. The Device Shadow 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 device's shadow:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "iot:DeleteThingShadow",  
            "Resource": ["arn:aws:iot:region:account:thing/thing"]  
        }  
    ]  
}
```

## Shadow MQTT Topics

The Device Shadow service uses reserved MQTT topics to enable applications and devices to get, update, or delete the state information for a device (shadow). The names of these topics start with `$aws/things/thingName/shadow`. Publishing and subscribing on shadow topics requires topic-based authorization. AWS IoT reserves the right to add new topics to the existing topic structure. For this reason, we recommend that you avoid wild card subscriptions to shadow topics. For example, avoid subscribing to topic filters like `$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 [Device Shadow Service Data Flow \(p. 348\)](#).

The following are the MQTT topics used for interacting with shadows.

### Topics

- [/update \(p. 370\)](#)
- [/update/accepted \(p. 371\)](#)
- [/update/documents \(p. 372\)](#)
- [/update/rejected \(p. 372\)](#)
- [/update/delta \(p. 373\)](#)
- [/get \(p. 374\)](#)
- [/get/accepted \(p. 374\)](#)
- [/get/rejected \(p. 375\)](#)
- [/delete \(p. 375\)](#)
- [/delete/accepted \(p. 376\)](#)
- [/delete/rejected \(p. 376\)](#)

## /update

Publish a request state document to this topic to update the device's 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 device updating its 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 \(p. 371\)](#) or [/update/rejected \(p. 372\)](#).

For more information, see [Request State Documents \(p. 377\)](#).

## 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 device's shadow:

```
$aws/things/thingName/shadow/update/accepted
```

For more information, see [Response State Documents \(p. 377\)](#).

## Example Policy

The following is an example of the required policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Subscribe"],
            "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/update/accepted"]
        }
    ]
}
```

```
        },
        {
            "Effect": "Allow",
            "Action": ["iot:Receive"],
            "Resource": ["arn:aws:iot:region:account:topic/$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 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"],
            "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/update/documents"]
        },
        {
            "Effect": "Allow",
            "Action": ["iot:Receive"],
            "Resource": ["arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update/accepted"]
        }
    ]
}
```

## /update/rejected

AWS IoT publishes an error response document to this topic when it rejects a change for the device's shadow:

```
$aws/things/thingName/shadow/update/rejected
```

For more information, see [Error Response Documents \(p. 379\)](#).

## Example Policy

The following is an example of the required policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Subscribe"],
            "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/update/rejected"]
        },
        {
            "Effect": "Allow",
            "Action": ["iot:Receive"],
            "Resource": ["arn:aws:iot:region:account:topic/$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 device's 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 \(p. 377\)](#).

## 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"],
            "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/update/delta"]
        },
        {
            "Effect": "Allow",
            "Action": ["iot:Receive"],
            "Resource": ["arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update/delta"]
        }
    ]
}
```

```
        ]
    }
```

## /get

Publish an empty message to this topic to get the device's shadow:

```
$aws/things/thingName/shadow/get
```

AWS IoT responds by publishing to either [/get/accepted \(p. 374\)](#) or [/get/rejected \(p. 375\)](#).

## 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 device's shadow:

```
$aws/things/thingName/shadow/get/accepted
```

For more information, see [Response State Documents \(p. 377\)](#).

## Example Policy

The following is an example of the required policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [ "iot:Subscribe" ],
            "Resource": [ "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/get/accepted" ]
        },
        {
            "Effect": "Allow",
            "Action": [ "iot:Receive" ],
            "Resource": [ "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/get/accepted" ]
        }
    ]
}
```

## /get/rejected

AWS IoT publishes an error response document to this topic when it can't return the device's shadow:

```
$aws/things/thingName/shadow/get/rejected
```

For more information, see [Error Response Documents \(p. 379\)](#).

## Example Policy

The following is an example of the required policy:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["iot:Subscribe"],  
            "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/get/rejected"]  
        },  
        {  
            "Action": ["iot:Receive"],  
            "Resource": ["arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/get/rejected"]  
        }  
    ]  
}
```

## /delete

To delete a device's 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 \(p. 376\)](#) or [/delete/rejected \(p. 376\)](#).

## Example Policy

The following is an example of the required policy:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["iot:Subscribe"],  
            "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/delete"]  
        },  
        {  
            "Effect": "Allow",  
            "Action": ["iot:Receive"],  
            "Resource": ["arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/delete"]  
        }  
    ]  
}
```

```
        ]  
    }
```

## /delete/accepted

AWS IoT publishes a message to this topic when a device's 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"],  
            "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/  
delete/accepted"]  
        },  
        {  
            "Effect": "Allow",  
            "Action": ["iot:Receive"],  
            "Resource": ["arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/delete/  
accepted"]  
        }  
    ]  
}
```

## /delete/rejected

AWS IoT publishes an error response document to this topic when it can't delete the device's shadow:

```
$aws/things/thingName/shadow/delete/rejected
```

For more information, see [Error Response Documents \(p. 379\)](#).

### Example Policy

The following is an example of the required policy:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["iot:Subscribe"],  
            "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/  
delete/rejected"]  
        },  
        {  
            "Effect": "Allow",  
            "Action": ["iot:Receive"],  
            "Resource": ["arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/rejected"]  
        }  
    ]  
}
```

```
        "Resource": [ "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/delete/  
rejected" ]  
    }  
}
```

## Shadow Document Syntax

The Device Shadow service uses the following documents in UPDATE, GET, and DELETE operations using the [RESTful API \(p. 367\)](#) or [MQTT Pub/Sub Messages \(p. 370\)](#). For more information, see [Device Shadow Service Documents \(p. 355\)](#).

### Examples

- [Request State Documents \(p. 377\)](#)
- [Response State Documents \(p. 377\)](#)
- [Error Response Documents \(p. 379\)](#)

## Request State Documents

Request state documents have the following format:

```
{  
    "state": {  
        "desired": {  
            "attribute1integer2,  
            "attribute2            ...  
            "attributeN        },  
        "reported": {  
            "attribute1integer1,  
            "attribute2            ...  
            "attributeN        }  
    },  
    "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 Device Shadow 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.
- **delta** — Only present if a thing reported different data in the `reported` and `desired` sections of the thing shadow 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 device's 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 [Shadow Error Messages \(p. 379\)](#).

## Shadow Error Messages

The Device Shadow 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)       | <ul style="list-style-type: none"> <li>• Invalid JSON</li> <li>• Missing required node: state</li> <li>• State node must be an object</li> <li>• Desired node must be an object</li> <li>• Reported node must be an object</li> <li>• Invalid version</li> <li>• Invalid clientToken</li> </ul> <p><b>Note</b><br/>A client token that is longer than 64 bytes will cause this response.</p> <ul style="list-style-type: none"> <li>• JSON contains too many levels of nesting; maximum is 6</li> <li>• State contains an invalid node</li> </ul> |
| 401 (Unauthorized)      | <ul style="list-style-type: none"> <li>• Unauthorized</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 403 (Forbidden)         | <ul style="list-style-type: none"> <li>• Forbidden</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 404 (Not Found)         | <ul style="list-style-type: none"> <li>• Thing not found</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 409 (Conflict)          | <ul style="list-style-type: none"> <li>• Version conflict</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 413 (Payload Too Large) | <ul style="list-style-type: none"> <li>• The payload exceeds the maximum size allowed</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

| HTTP Error Code              | Error Messages                                                                                                                                             |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 415 (Unsupported Media Type) | <ul style="list-style-type: none"><li>Unsupported documented encoding; supported encoding is UTF-8</li></ul>                                               |
| 429 (Too Many Requests)      | <ul style="list-style-type: none"><li>The Device Shadow service will generate this error message when there are more than 10 in-flight requests.</li></ul> |
| 500 (Internal Server Error)  | <ul style="list-style-type: none"><li>Internal service failure</li></ul>                                                                                   |

# Jobs

AWS IoT jobs can be used to define a set of remote operations that are sent to and executed on one or more devices connected to AWS IoT.

## Tip

For job document examples, see the [jobs-agent.js](#) example in the AWS IoT SDK for JavaScript.

## Jobs Key Concepts

### job

A job is a remote operation that is 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.

### job document

To create a job, you must first create a job document that is a description of the remote operations to be performed by the devices.

Job documents are UTF-8 encoded JSON documents and should contain information that your devices need to perform a job. A job document contains one or more URLs where the device can download an update or some other data. The job document can be stored in an Amazon S3 bucket, or be included inline with the command that creates the job.

### target

When you create a job, you specify a list of targets that are the devices that should perform the operations. The targets can be things or [thing groups \(p. 99\)](#) or both. The AWS IoT Jobs service sends a message to each target to inform it that a job is available.

### job execution

A job execution is an instance of a job on a target device. The target starts an execution of a job by downloading the job document. It then performs the operations specified in the document, and reports its progress to AWS IoT. An execution number is a unique identifier of a job execution on a specific target. The Jobs service provides commands to track the progress of a job execution on a target and the progress of a job across all targets.

### snapshot job

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.

### continuous job

A continuous job is sent to all targets that you specify when you create the job. It continues to run and is sent to any new devices (things) that are added to the target group. For example, a continuous job can be used to onboard or upgrade devices as they are added to a group. You can make a job continuous by setting an optional parameter when you create the job.

### rollouts

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.

The following field can be added to the `CreateJob` request to specify the maximum number of job targets to inform per minute. This example sets a static rollout rate.

```
"jobExecutionRolloutConfig": {  
    "maximumPerMinute": "integer"  
}
```

You can also set a variable rollout rate with the `exponentialRate` field. The following example creates a rollout that has an exponential rate.

```
"jobExecutionsRolloutConfig": {  
    "exponentialRate": {  
        "baseRatePerMinute": integer,  
        "incrementFactor": integer,  
        "rateIncreaseCriteria": {  
            "numberOfNotifiedThings": integer, // Set one or the other  
            "numberOfSucceededThings": integer // of these two values.  
        },  
        "maximumPerMinute": integer  
    }  
}
```

For more information about configuring job rollouts, see [Job Rollout and Abort Configuration](#).

#### abort

You can create a set of conditions to abort rollouts when criteria that you specify have been met. For more information, see [Job Rollout and Abort Configuration](#).

#### presigned URLs

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.

In the Beijing and Ningxia Regions, presigned URLs work only if the resource owner has an ICP license. For more information, see [Amazon Simple Storage Service](#) in the [Getting Started with AWS Services in China](#) documentation.

#### timeouts

Job timeouts make it possible to be notified whenever a job execution gets stuck in the `IN_PROGRESS` state for an unexpectedly long period of time. There are two types of timers: in-progress timers and step timers.

When you create a job, you can set a value for the `inProgressTimeoutInMinutes` property of the optional `TimeoutConfig` object. The in-progress timer can't be updated and applies to all job executions for the job. Whenever a job execution remains in the `IN_PROGRESS` status for longer

than this interval, the job execution fails and switches to the terminal `TIMED_OUT` status. AWS IoT also publishes an MQTT notification.

You can also set a step timer for a job execution by setting a value for `stepTimeoutInMinutes` when you call [UpdateJobExecution](#). The step timer applies only to the job execution that you update. You can set a new value for this timer each time you update a job execution. You can also create a step timer when you call [StartNextPendingJobExecution](#). If the job execution remains in the `IN_PROGRESS` status for longer than the step timer interval, it fails and switches to the terminal `TIMED_OUT` status. The step timer has no effect on the in-progress timer that you set when you create a job.

The following diagram and description illustrate the ways in which in-progress timeouts and step timeouts interact with each other.

**Job Creation:** `CreateJob` sets an in-progress timer that expires in twenty minutes. This timer applies to all job executions and can't be updated.

**12:00 PM:** The job execution starts and switches to `IN_PROGRESS` status. The in-progress timer starts to run.

**12:05 PM:** `UpdateJobExecution` creates a step timer with a value of 7 minutes. If a new step timer isn't created, the job execution times out at 12:12 PM.

**12:10 PM:** `UpdateJobExecution` creates a new step timer with a value of 5 minutes. The previous step timer is discarded. If a new step timer isn't created, the job execution times out at 12:15 PM.

**12:13 PM:** `UpdateJobExecution` creates a new step timer with a value of 9 minutes. The job execution times out at 12:20 because the in-progress timer expires at 12:20. The step timer can't exceed the absolute bound created by the in-progress timer.

`UpdateJobExecution` can also discard a step timer that has already been created by creating a new step timer with a value of -1.

## Managing Jobs

You can use the [AWS IoT console](#), the Jobs HTTPS API, the AWS Command Line Interface, or the AWS SDKs to create and manage jobs. For more information, see [Job Management and Control API \(p. 405\)](#), [AWS CLI Command Reference: iot](#) or [AWS SDKs and Tools](#).

The primary purpose of jobs is to notify devices of a software or firmware update. When sending code to devices, the best practice is to sign the code file. This allows devices to detect if the code has been modified in transit. The instructions in the following section are written with the assumption that you want to code-sign the software update you are sending to your devices.

For more information, see [What Is Code Signing for AWS IoT?](#)

Before you create a job, you must create a job document. If you are using Code-signing for AWS IoT, you must upload your job document to a versioned Amazon S3 bucket. For more information about creating an Amazon S3 bucket and uploading files to it, see [Getting Started with Amazon Simple Storage Service](#) in the [Amazon S3 Getting Started Guide](#).

Your job document can contain a presigned Amazon S3 URL that points to your code file (or other file). Presigned Amazon S3 URLs are valid for a limited amount of time and so are not generated until a device requests a job document. Because the presigned URL has not been created when you are creating the job document, you put a placeholder URL in your job document instead.

A placeholder URL looks like the following: \${aws:iot:s3-presigned-url:https://s3.region.amazonaws.com/<bucket>/<code file>} where **bucket** is the Amazon S3 bucket that contains the code file and **code file** is the Amazon S3 key of the code file.

When a device requests the job document, AWS IoT generates the presigned URL and replaces the placeholder URL with the presigned URL. Your job document is then sent to the device.

When you create a job that uses presigned Amazon S3 URLs, you must provide an IAM role 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.

You can specify an optional timeout for the presigned URL. For more information, see [CreateJob \(p. 420\)](#).

### To grant the Jobs service 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 navigation pane, choose **Roles**.
3. Search for your role and choose it.
4. On the **Trust Relationships** tab, choose **Edit Trust Relationship**.
5. 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"  
    }  
  ]  
}
```

6. Choose **Update Trust Policy**.
7. If your job uses a job document that is an Amazon S3 object, choose **Permissions** and with the following JSON, add a policy that grants permission to download files from your Amazon S3 bucket:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "s3:GetObject",  
      "Resource": "arn:aws:s3:::your_S3_bucket/*"  
    }  
  ]  
}
```

## Creating and Managing Jobs (Console)

If you are using Code-signing for AWS IoT, you must add two placeholder URLs in your job document:

A placeholder for the code file should look like this: \${aws:iot:s3-presigned-url:https://s3.amazonaws.com/<my-s3-bucket>/<my-code-file>}.

**Note**

Currently, versions are not supported for presigned URL placeholders for jobs. If you update your code file and copy it to the same Amazon S3 location, you must create a new signature and then reference the new signature version in your job document.

A placeholder for the signature should look like this: \${aws:iot:code-sign-signature:s3://<region>.<my-s3-bucket>/<my-code-file>@<code-file-version-id>}.

**To create a job**

1. Browse to the [AWS IoT console](#).
2. In the navigation pane, choose **Manage**, and then choose **Jobs**.
3. Choose **Create a job**.
4. Choose **Create a custom job**.
5. Enter an alphanumeric ID for your job and an optional description.

**Note**

We do not recommend using personally identifiable information in your job IDs or descriptions.

6. Select the device or device groups that you want to update.
7. Under **Add a job file**, choose **Select**, and then select your job document.
8. Choose **Sign image for me**. If you are not code signing your update, you can skip this step.
9. Create or choose a code-signing profile. If you are not code signing your update, you can skip this step.
10. Under **Pre-sign resource URLs**, choose **I want to pre-sign my URLs and have configured my job file**. If you are not code signing your update, you can skip this step.
11. Choose a role and an expiry time for the presigned URL.
12. Under **Job type**, choose the appropriate option for your update, and then choose **Next**.
13. Specify values for any advanced configurations, and then choose **Create**.

After you create the job, the console generates a JSON signature and places it in your job document.

You can use the [AWS IoT console](#) to view the status, cancel, or delete a job.

1. Browse to the [AWS IoT console](#).
2. In the navigation pane, choose **Manage**, and then choose **Jobs**.

## Creating and Managing Jobs (CLI)

This section describes how to create and manage jobs.

### Create Jobs

You use the **CreateJob** command to create an AWS IoT job. The job is queued for execution on the targets (things or thing groups) that you specify. To create an AWS IoT 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 the AWS IoT Jobs service to assume the role.

## Code-signing with Jobs

If you are using Code-signing for AWS IoT, you must start a code-signing job and include the output in your job document. Use the [start-signing-job](#) command to create a code-signing job. `start-signing-job` returns a job ID. Use the [describe-signing-job](#) command to get the Amazon S3 location where the signature is stored. You can then download the signature from Amazon S3. For more information about code signing jobs, see [Code-signing for AWS IoT](#).

Your job document must contain a presigned URL placeholder for your code file and the JSON signature output placed in an Amazon S3 bucket using the `start-signing-job` command, enclosed in a `codesign` element:

```
{
    "presign": "${aws:iot:s3-presigned-url:https://s3.region.amazonaws.com/bucket/image}",
    "codesign": {
        "rawPayloadSize": <image-file-size>,
        "signature": <signature>,
        "signatureAlgorithm": <signature-algorithm>,
        "payloadLocation": {
            "s3": {
                "bucketName": <my-s3-bucket>,
                "key": <my-code-file>,
                "version": <code-file-version-id>
            }
        }
    }
}
```

## Creating a Job with a Job Document

The following command shows how to create a job using a job document (`job-document.json`) stored in an Amazon S3 bucket (`jobBucket`) and a role with permission to download files from Amazon S3 (`S3DownloadRole`).

```
aws iot create-job \
    --job-id 010 \
    --targets arn:aws:iot:us-east-1:123456789012:thing/thingOne \
    --document-source https://s3.amazonaws.com/my-s3-bucket/job-document.json \
    --timeout-config inProgressTimeoutInMinutes=100 \
    --job-executions-rollout-config "{\"exponentialRate\": { \"baseRatePerMinute\": 50, \
    \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings\": 1000, \
    \"numberOfSucceededThings\": 1000}, \"maximumPerMinute\": 1000}}\" \
    --abort-config "{\"criterialist\": [ { \"action\": \"CANCEL\", \"failureType\": \
    \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20}, { \"action\": \
    \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings\": 200, \
    \"thresholdPercentage\": 50}]}\" \
    --presigned-url-config "{\"roleArn\": \"arn:aws:iam::123456789012:role/ \
    S3DownloadRole\", \"expiresInSec\": 3600}"
```

The job is executed on `thingOne`.

The optional `timeout-config` parameter specifies the amount of time each device has to finish its execution of the job. The timer starts when the job execution status is set to `IN_PROGRESS`. If the job execution status is not set to another terminal state before the time expires, it is set to `TIMED_OUT`.

The in-progress timer can't be updated and applies to all job executions for the job. Whenever a job execution remains in the `IN_PROGRESS` state for longer than this interval, the job execution fails and switches to the terminal `TIMED_OUT` status. AWS IoT also publishes an MQTT notification.

For more information about creating configurations about job rollouts and aborts, see [Job Rollout and Abort Configuration](#).

### 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.

## Update a Job

You use the **UpdateJob** command to update a job. You can update the `description`, `presignedUrlConfig`, `jobExecutionsRolloutConfig`, `abortConfig`, and `timeoutConfig` fields of a job.

```
aws iot update-job \
--job-id 010 \
--description "updated description" \
--timeout-config inProgressTimeoutInMinutes=100 \
--job-executions-rollout-config "{\"exponentialRate\": { \"baseRatePerMinute\": 50,
\"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings\": 1000,
\"numberOfSucceededThings\": 1000}, \"maximumPerMinute\": 1000}}" \
--abort-config "{\"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType\": \"FAILED\",
\"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20}, { \"action\": \"CANCEL\",
\"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings\": 200,
\"thresholdPercentage\": 50}]}"
--presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/S3DownloadRole\",
\"expiresInSec\":3600}
```

For more information, see [Job Rollout and Abort Configuration](#).

## Cancel a Job

You use the **CancelJob** command to cancel a job. Canceling a job stops AWS IoT from rolling out any new job executions for the job. It also cancels any job executions that are in a `QUEUED` state. AWS IoT leaves any job executions in a terminal state untouched because the device has already completed the job. If the status of a job execution is `IN_PROGRESS`, it also remains untouched unless you use the optional `--force` parameter.

The following command shows how to cancel a job with ID 010.

```
aws iot cancel-job --job-id 010
```

The command displays the following output:

```
{
  "jobArn": "string",
  "jobId": "string",
  "description": "string"
}
```

When you cancel a job, job executions that are in a `QUEUED` state are canceled. Job executions that are in an `IN_PROGRESS` state are canceled if you specify the optional `--force` parameter. Job executions in a terminal state are not canceled.

### Warning

Canceling a job that is in the `IN_PROGRESS` state (by setting the `--force` parameter) cancels any job executions that are in progress and causes the device that is executing the job to be unable to update the job execution status. Use caution and make sure that each device executing a canceled job can recover to a valid state.

The status of a canceled job or of one of its job executions is eventually consistent. AWS IoT stops scheduling new job executions and `QUEUED` job executions for that job to devices as soon as possible. Changing the status of a job execution to `CANCELED` might take some time, depending on the number of devices and other factors.

If a job is canceled because it has met the criteria defined by an `AbortConfig` object, the service adds auto-populated values for the `comment` and `reasonCode` fields. You can create your own values for `reasonCode` when the job cancellation is user-driven.

## Cancel a Job Execution

You use the **CancelJobExecution** command to cancel a job execution on a device. It cancels a job execution that is in a `QUEUED` state. If you want to cancel a job execution that is in progress, you must use the `--force` parameter.

The following command shows how to cancel the job execution from job 010 running on `myThing`.

```
aws iot cancel-job-execution --job-id 010 --thing-name myThing
```

The command displays no output.

A job execution that is in a `QUEUED` state is canceled. A job execution that is in an `IN_PROGRESS` state is canceled if you specify the optional `--force` parameter. Job executions in a terminal state cannot be canceled.

### Warning

When you cancel a job execution that is in the `IN_PROGRESS` state, the device cannot update the job execution status. Use caution and ensure that the device can recover to a valid state.

If the job execution is in a terminal state or if the job execution is in an `IN_PROGRESS` state and the `--force` parameter is not set to `true`, this command causes an `InvalidStateTransitionException`.

The status of a canceled job execution is eventually consistent. Changing the status of a job execution to `CANCELED` might take some time, depending on various factors.

## Delete a Job

You use the **DeleteJob** command to delete a job and its job executions. By default, you can only delete a job that is in a terminal state (`SUCCEEDED` or `CANCELED`). Otherwise, an exception occurs. You can delete a job in the `IN_PROGRESS` state if the `force` parameter is set to `true`.

Run the following command to delete a job:

```
aws iot delete-job --job-id 010 --force|--no-force
```

The command displays no output.

### Warning

When you delete a job that is in the `IN_PROGRESS` state, the device that is executing the job cannot access job information or update the job execution status. Use caution and make sure that each device executing a job that has been deleted can recover to a valid state.

It can take some time to delete a job, depending on the number of job executions created for the job and other factors. While the job is being deleted, `DELETION_IN_PROGRESS` appears as the status of the job. An error results if you attempt to delete or cancel a job whose status is already `DELETION_IN_PROGRESS`.

Only 10 jobs can have a status of `DELETION_IN_PROGRESS` at the same time. Otherwise, a `LimitExceededException` occurs.

## Get a Job Document

You use the **GetJobDocument** command to retrieve a job document for a job. A job document is a description of the remote operations to be performed by the devices.

Run the following command to get a job document:

```
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/\nfirmWareUpdate-01\",\\n\t\"data\":\"\$\{aws:iot:s3-presigned-url:https://s3.amazonaws.com/job-\ntest-bucket/datafile}\\"\n}"
}
```

### Note

When you use this command to retrieve a job document, placeholder URLs are not replaced by presigned Amazon S3 URLs. When a device calls the [GetPendingJobExecutions \(p. 463\)](#) MQTT API, the placeholder URLs are replaced by presigned Amazon S3 URLs in the job document.

## List Jobs

You use the **ListJobs** command to get a list of all jobs in your AWS account. Job data and job execution data are retained for a [limited time](#). Run the following command to list all jobs in your AWS account:

```
aws iot list-jobs
```

The command returns 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": "SUCCEEDED",
            "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
        }
    ]
}
```

```
        "createdAt": 1486678850.575,
        "targetSelection": "SNAPSHOT",
        "jobId": "011"
    }
}
```

## Describe a Job

Run the **DescribeJob** command to get the status of a job. The following command shows how to describe a job:

```
$ aws iot describe-job --job-id 010
```

The command returns the status of the specified job. For example:

```
{
    "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,
            "numberOfTimedOutThings": 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",
        "jobExecutionsRolloutConfig": {
            "exponentialRate": {
                "baseRatePerMinute": integer,
                "incrementFactor": integer,
                "rateIncreaseCriteria": {
                    "numberOfNotifiedThings": integer, // Set one or the other
                    "numberOfSucceededThings": integer // of these two values.
                },
                "maximumPerMinute": integer
            }
        },
        "abortConfig": {
            "criteriaList": [
            {

```

```

        "action": "string",
        "failureType": "string",
        "minNumberOfExecutedThings": integer,
        "thresholdPercentage": integer
    }
]
},
"timeoutConfig": {
    "inProgressTimeoutInMinutes": number
}
}
}

```

## List Executions for a Job

A job running on a specific device is represented by a job execution object. Run the **ListJobExecutionsForJob** command to list all job executions for a job. The following shows how to list the executions for a job:

```
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

Run the **ListJobExecutionsForThing** command to list all job executions running on a thing. The following shows how to list job executions for a thing:

```
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_PROGRESS",
        "startAt": 1486685870.729,
        "lastUpdatedAt": 1486685870.729,
        "queuedAt": 1486685870.729,
        "executionNumber": 1357924680
    },
    "jobId": "012"
},
{
    "jobExecutionSummary": {
        "status": "SUCCEEDED",
        "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

Run the **DescribeJobExecution** command to get the status of a job execution. You must specify a job ID and thing name and, optionally, an execution number to identify the job execution. The following shows how to describe a job execution:

```
aws iot describe-job-execution --job-id 017 --thing-name thingOne
```

The command returns the [JobExecution \(p. 409\)](#). For example:

```
{
    "execution": {
        "jobId": "017",
        "executionNumber": 4516820379,
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingOne",
        "versionNumber": 123,
        "createdAt": 1489084805.285,
        "lastUpdatedAt": 1489086279.937,
        "startedAt": 1489086279.937,
        "status": "IN_PROGRESS",
        "currentPhase": "PENDING_ACCEPTANCE"
    }
}
```

```
        "approximateSecondsBeforeTimedOut": 100,
        "statusDetails": {
            "status": "IN_PROGRESS",
            "detailsMap": {
                "percentComplete": "10"
            }
        }
    }
}
```

## Delete Job Execution

Run the **DeleteJobExecution** command to delete a job execution. You must specify a job ID, a thing name, and an execution number to identify the job execution. The following shows how to delete a job execution:

```
aws iot delete-job-execution --job-id 017 --thing-name thingOne --execution-number
1234567890 --force|--no-force
```

The command displays no output.

By default, the status of the job execution must be `QUEUED` or in a terminal state (`SUCCEEDED`, `FAILED`, `REJECTED`, `TIMED_OUT`, `REMOVED` or `CANCELED`). Otherwise, an error occurs. To delete a job execution with a status of `IN_PROGRESS`, you can set the `force` parameter to `true`.

### Warning

When you delete a job execution with a status of `IN_PROGRESS`, the device that is executing the job cannot access job information or update the job execution status. Use caution and make sure that the device can recover to a valid state.

# Devices and Jobs

## Device Communication with Jobs

Devices can communicate with the AWS IoT Jobs service through these methods:

- MQTT
- HTTP Signature Version 4
- HTTP TLS

### Using the MQTT Protocol

Communication between the AWS IoT Jobs service and your devices can occur over the MQTT protocol. Devices subscribe to MQTT topics to be notified of new jobs and to receive responses from the AWS IoT Jobs service. Devices publish on MQTT topics to query or update the state of a job execution. Each device has its own general MQTT topic. For more information about publishing and subscribing to MQTT topics, see [Message Broker for AWS IoT \(p. 244\)](#).

### Note

You must use the correct endpoint when you communicate with the AWS IoT Jobs service through MQTT. Use the **DescribeEndpoint** command to find it. For example, if you run this command:

```
aws iot describe-endpoint --endpoint-type iot:Data
```

you get a result similar to the following:

```
{  
    "endpointAddress": "a1b2c3d4e5f6g7.iot.us-west-2.amazonaws.com"  
}
```

With this method, your device uses its device-specific certificate and private key to authenticate with the AWS IoT Jobs service.

Devices can:

- Be notified when a job execution is added or removed from the list of pending job executions 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 \(p. 477\)](#) API.
- Query the status of a job execution by calling the [DescribeJobExecution \(p. 472\)](#) API.
- Retrieve a list of pending job executions by calling the [GetPendingJobExecutions \(p. 463\)](#) API.
- Retrieve the next pending job execution by calling the [DescribeJobExecution \(p. 472\)](#) API with `jobId $next`.
- Get and start the next pending job execution by calling the [StartNextPendingJobExecution \(p. 467\)](#) API.

The AWS IoT Jobs service publishes success and failure messages on an MQTT topic. The topic is formed by appending `accepted` or `rejected` to the topic used to make the request. For example, if a request message is published on the `$aws/things/myThing/jobs/get` topic, the AWS IoT Jobs service publishes success messages on the `$aws/things/myThing/jobs/get/accepted` topic and publishes rejected messages on the `$aws/things/myThing/jobs/get/rejected` topic.

#### Using HTTP Signature Version 4

Communication between the AWS IoT Jobs service and your devices can occur over HTTP Signature Version 4 on port 443. This is the method used by the AWS SDKs and CLI. For more information about those tools, see [AWS CLI Command Reference: iot-jobs-data](#) or [AWS SDKs and Tools](#) and refer to the `iotJobsDataPlane` section for your preferred language.

##### Note

You must use the correct endpoint when you communicate with the AWS IoT Jobs service through HTTP Signature Version 4 or using an AWS SDK or CLI `iotJobsDataPlane` command. Use the `DescribeEndpoint` command to find it. For example, if you run this command:

```
aws iot describe-endpoint --endpoint-type iot:Jobs
```

you get a result similar to the following:

```
{  
    "endpointAddress": "a1b2c3d4e5f6g7.jobs.iot.us-west-2.amazonaws.com"  
}
```

With this method of communication, your device uses IAM credentials to authenticate with the AWS IoT Jobs service.

The following commands are available using this method:

- **DescribeJobExecution**

```
aws iot-jobs-data describe-job-execution ...
```

- **GetPendingJobExecutions**

```
aws iot-jobs-data get-pending-job-executions ...
```

- **StartNextPendingJobExecution**

```
aws iot-jobs-data start-next-pending-job-execution ...
```

- **UpdateJobExecution**

```
aws iot-jobs-data update-job-execution ...
```

### Using HTTP TLS

Communication between the AWS IoT Jobs service and your devices can occur over HTTP TLS on port 8443 using a third-party software client that supports this protocol.

#### Note

You must use the correct endpoint when you communicate with the AWS IoT Jobs service through HTTP TLS. Use the **DescribeEndpoint** command to find it. For example, if you run this command:

```
aws iot describe-endpoint --endpoint-type iot:Jobs
```

you get a result similar to the following:

```
{  
    "endpointAddress": "a1b2c3d4e5f6g7.jobs.iot.us-west-2.amazonaws.com"  
}
```

With this method, your device uses X.509 certificate-based authentication (for example, its device-specific certificate and private key).

The following commands are available using this method:

- **DescribeJobExecution**
- **GetPendingJobExecutions**
- **StartNextPendingJobExecution**
- **UpdateJobExecution**

## Programming Devices to Work with Jobs

The examples in this section use MQTT to illustrate how a device works with the AWS IoT Jobs service. Alternatively, you could use the corresponding API or CLI commands. For these examples, we assume a device called *MyThing* subscribes 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`

If you are using Code-signing for AWS IoT your device code must verify the signature of your code file. The signature is in the job document in the `codesign` property. For more information about verifying a code file signature, see [Device Agent Sample](#).

## Device Workflow

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 \(p. 472\)](#) MQTT API with `jobId $next` to get the next job, its job document, and other details, including any state saved in `statusDetails`. If the job document has a code file signature, you must verify the signature before proceeding with processing the job request.
3. Call the [UpdateJobExecution \(p. 477\)](#) MQTT API to update the job status. Or, to combine this and the previous step in one call, the device can call [StartNextPendingJobExecution \(p. 467\)](#).
4. (Optional) You can add a step timer by setting a value for `stepTimeoutInMinutes` when you call either [UpdateJobExecution \(p. 477\)](#) or [StartNextPendingJobExecution \(p. 467\)](#).
5. Perform the actions specified by the job document using the [UpdateJobExecution \(p. 477\)](#) MQTT API to report on the progress of the job.
6. Continue to monitor the job execution by calling the [DescribeJobExecution \(p. 472\)](#) MQTT API with this `jobId`. If the job execution is canceled or deleted while the device is running the job, the device should be capable of recovering to a valid state.
7. Call the [UpdateJobExecution \(p. 477\)](#) MQTT API when finished with the job to update the job status and report success or failure.
8. Because this job's execution status has been changed to a terminal state, the next job available for execution (if any) changes. 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 continues to receive notifications of the next pending job execution, including its job execution data, when it completes a job or 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 \(p. 463\)](#) 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 \(p. 472\)](#) MQTT API to get the job document and other details, including any state saved in `statusDetails`.
5. Call the [UpdateJobExecution \(p. 477\)](#) 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. Optionally, you can add a step timer by setting a value for `stepTimeoutInMinutes` when you call [UpdateJobExecution \(p. 477\)](#).
7. Perform the actions specified by the job document using the [UpdateJobExecution \(p. 477\)](#) MQTT API to report on the progress of the job.
8. Continue to monitor the job execution by calling the [DescribeJobExecution \(p. 472\)](#) MQTT API with this `jobId`. If the job execution is canceled or deleted while the device is running the job, the device should be capable of recovering to a valid state.
9. Call the [UpdateJobExecution \(p. 477\)](#) MQTT API when finished with the job to update the job status and to report success or failure.

If the device remains online, it is notified of all pending job executions when 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 \(p. 477\)](#) MQTT API to update the job status to REJECTED.

## Starting a New Job

### new job notification

When a new job is created, the AWS IoT Jobs service publishes a message on the \$aws/things/*thing-name*/jobs/notify topic for each target device.

#### More Information(1)

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 \(p. 472\)](#) MQTT API with the includeJobDocument field set to true (the default).

#### More Information(2)

If the request is successful, the AWS IoT Jobs service publishes a message on the \$aws/things/MyThing/jobs/0023/get/accepted topic:

```
{  
    "clientToken" : "client-001",  
    "timestamp" : 1489097434407,  
    "execution" : {  
        "approximateSecondsBeforeTimedOut": number,  
        "jobId" : "023",  
        "status" : "QUEUED",  
        "queuedAt" : 1489097374841,  
        "lastUpdatedAt" : 1489097374841,  
        "versionNumber" : 1,  
        "jobDocument" : {  
            < contents of job document >  
        }  
    }  
}
```

#### Note

If the request fails, the AWS IoT Jobs service publishes a message on the \$aws/things/MyThing/jobs/0023/get/rejected topic.

The device now has the job document that it can use 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.

## Report Job Execution Status

### Update Execution Status

As the device is executing the job, it can call the [UpdateJobExecution \(p. 477\)](#) MQTT API to update the status of the job execution.

#### More Information (3)

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 \(p. 467\)](#). That 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.

If the job contains a `TimeoutConfig`, the in-progress timer starts running. You can also set a step timer for a job execution by setting a value for `stepTimeoutInMinutes` when you call [UpdateJobExecution](#). The step timer applies only to the job execution that you update. You can set a new value for this timer each time you update a job execution. You can also create a step timer when you call [StartNextPendingJobExecution](#). If the job execution remains in the `IN_PROGRESS` status for longer than the step timer interval, it fails and switches to the terminal `TIMED_OUT` status. The step timer has no effect on the in-progress timer that you set when you create a job.

The `status` field can be set to `IN_PROGRESS`, `SUCCEEDED`, or `FAILED`. 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 \(p. 477\)](#) MQTT API. If the job was successful, set `status` to `SUCCEEDED` and, in the message payload, in `statusDetails`, add other information about the job as name-value pairs. The in-progress and step timers end when the job execution is complete.

#### More Information(4)

For example:

```
{
    "status": "SUCCEEDED",
    "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 the AWS IoT Jobs service 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 Information(5)

For example:

```
{
    "timestamp": 1476290692776,
    "jobs": {
        "QUEUED": [
            {
                "jobId": "0002",
                "queuedAt": 1476290646230,
                "lastUpdatedAt": 1476290646230
            }
        ],
        "IN_PROGRESS": [
            {
                "jobId": "0003",
                "queuedAt": 1476290646230,
                "lastUpdatedAt": 1476290646230
            }
        ]
    }
}
```

## Jobs Notifications

The AWS IoT Jobs service publishes MQTT messages to reserved topics when jobs are pending or when the first job execution in the list changes. Devices can keep track of pending jobs by subscribing to these topics.

Job notifications are published to MQTT topics as JSON payloads. There are two kinds of notifications:

- A `ListNotification` contains a list of no more than 10 pending job executions. The job executions in this list have status values of either `IN_PROGRESS` or `QUEUED`. They are sorted by status (`IN_PROGRESS` job executions before `QUEUED` job executions) and then by the times when they were queued.

A `ListNotification` is published whenever one of the following criteria is met.

- A new job execution is queued or changes to a non-terminal status (`IN_PROGRESS` or `QUEUED`).
- An old job execution changes to a terminal status (`FAILED`, `SUCCEEDED`, `CANCELED`, `TIMED_OUT`, `REJECTED`, or `MOVED`).
- A `NextNotification` contains summary information about the one job execution that is next in the queue.

A `NextNotification` is published whenever the first job execution in the list changes.

- A new job execution is added to the list as `QUEUED`, and it is the first one in the list.
- The status of an existing job execution that was not the first one in the list changes from `QUEUED` to `IN_PROGRESS` and becomes the first one in the list. (This happens when there are no other `IN_PROGRESS` job executions in the list or when the job execution whose status changes from `QUEUED` to `IN_PROGRESS` was queued earlier than any other `IN_PROGRESS` job execution in the list.)
- The status of the job execution that is first in the list changes to a terminal status and is removed from the list.

For more information about publishing and subscribing to MQTT topics, see [Message Broker for AWS IoT \(p. 244\)](#).

### Note

Notifications are not available when you use HTTP Signature Version 4 or HTTP TLS to communicate with jobs.

job pending

The AWS IoT Jobs service 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 the first job execution in the list changes:

- `$aws/things/thingName/jobs/notify`
- `$aws/things/thingName/jobs/notify-next`

### More Information(6)

The messages contain the following example payloads:

`$aws/things/thingName/jobs/notify:`

```
{  
    "timestamp" : 10011,  
    "jobs" : {  
        "IN_PROGRESS" : [ {  
            "jobId" : "other-job",  
            "queuedAt" : 10003,  
            "lastUpdatedAt" : 10009,
```

```

        "executionNumber" : 1,
        "versionNumber" : 1
    } ],
    "QUEUED" : [ {
        "jobId" : "this-job",
        "queuedAt" : 10011,
        "lastUpdatedAt" : 10011,
        "executionNumber" : 1,
        "versionNumber" : 0
    } ]
}
}

```

\$aws/things/*thingName*/jobs/notify-next:

```

{
    "timestamp" : 10011,
    "execution" : {
        "jobId" : "other-job",
        "status" : "IN_PROGRESS",
        "queuedAt" : 10009,
        "lastUpdatedAt" : 10009,
        "versionNumber" : 1,
        "executionNumber" : 1,
        "jobDocument" : {"c":"d"}
    }
}

```

Possible job execution status values are QUEUED, IN\_PROGRESS, FAILED, SUCCEEDED, CANCELED, TIMED\_OUT, REJECTED, and REMOVED.

The following series of examples show the notifications that are published to each topic as job executions are created and change from one state to another.

First, one job, called *job1*, is created. This notification is published to the jobs/notify topic:

```

{
    "timestamp": 1517016948,
    "jobs": {
        "QUEUED": [
            {
                "jobId": "job1",
                "queuedAt": 1517016947,
                "lastUpdatedAt": 1517016947,
                "executionNumber": 1,
                "versionNumber": 1
            }
        ]
    }
}

```

This notification is published to the jobs/notify-next topic:

```

{
    "timestamp": 1517016948,
    "execution": {
        "jobId": "job1",
        "status": "QUEUED",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517016947,
        "versionNumber": 1,
        "executionNumber": 1,
}

```

```

        "jobDocument": {
            "operation": "test"
        }
    }
}

```

When another job is created (`job2`), this notification is published to the `jobs/notify` topic:

```

{
    "timestamp": 1517017192,
    "jobs": {
        "QUEUED": [
            {
                "jobId": "job1",
                "queuedAt": 1517016947,
                "lastUpdatedAt": 1517016947,
                "executionNumber": 1,
                "versionNumber": 1
            },
            {
                "jobId": "job2",
                "queuedAt": 1517017191,
                "lastUpdatedAt": 1517017191,
                "executionNumber": 1,
                "versionNumber": 1
            }
        ]
    }
}

```

A notification is not published to the `jobs/notify-next` topic because the next job in the queue (`job1`) has not changed. When `job1` starts to execute, its status changes to `IN_PROGRESS`. No notifications are published because the list of jobs and the next job in the queue have not changed.

When a third job (`job3`) is added, this notification is published to the `jobs/notify` topic:

```

{
    "timestamp": 1517017906,
    "jobs": {
        "IN_PROGRESS": [
            {
                "jobId": "job1",
                "queuedAt": 1517016947,
                "lastUpdatedAt": 1517017472,
                "startedAt": 1517017472,
                "executionNumber": 1,
                "versionNumber": 2
            }
        ],
        "QUEUED": [
            {
                "jobId": "job2",
                "queuedAt": 1517017191,
                "lastUpdatedAt": 1517017191,
                "executionNumber": 1,
                "versionNumber": 1
            },
            {
                "jobId": "job3",
                "queuedAt": 1517017905,
                "lastUpdatedAt": 1517017905,
                "executionNumber": 1,
                "versionNumber": 1
            }
        ]
    }
}

```

```

        }
    ]
}
}
```

A notification is not published to the `jobs/notify-next` topic because the next job in the queue is still `job1`.

When `job1` is complete, its status changes to `SUCCEEDED`, and this notification is published to the `jobs/notify` topic:

```

{
  "timestamp": 1517186269,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517017905,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

At this point, `job1` has been removed from the queue, and the next job to be executed is `job2`. This notification is published to the `jobs/notify-next` topic:

```

{
  "timestamp": 1517186269,
  "execution": {
    "jobId": "job2",
    "status": "QUEUED",
    "queuedAt": 1517017191,
    "lastUpdatedAt": 1517017191,
    "versionNumber": 1,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
```

If `job3` must begin executing before `job2` (which is not recommended), the status of `job3` can be changed to `IN_PROGRESS`. If this happens, `job2` is no longer next in the queue, and this notification is published to the `jobs/notify-next` topic:

```

{
  "timestamp": 1517186779,
  "execution": {
    "jobId": "job3",
    "status": "IN_PROGRESS",
    "queuedAt": 1517017905,
```

```
        "startedAt": 1517186779,
        "lastUpdatedAt": 1517186779,
        "versionNumber": 2,
        "executionNumber": 1,
        "jobDocument": {
            "operation": "test"
        }
    }
}
```

No notification is published to the `jobs/notify` topic because no job has been added or removed.

If the device rejects job2 and updates its status to `REJECTED`, this notification is published to the `jobs/notify` topic:

```
{
    "timestamp": 1517189392,
    "jobs": {
        "IN_PROGRESS": [
            {
                "jobId": "job3",
                "queuedAt": 1517017905,
                "lastUpdatedAt": 1517186779,
                "startedAt": 1517186779,
                "executionNumber": 1,
                "versionNumber": 2
            }
        ]
    }
}
```

If job3 (which is still in progress) is force deleted, this notification is published to the `jobs/notify` topic:

```
{
    "timestamp": 1517189551,
    "jobs": {}
}
```

At this point, the queue is empty. This notification is published to the `jobs/notify-next` topic:

```
{
    "timestamp": 1517189551
}
```

## Using the AWS IoT Jobs APIs

There are two categories of API used in the AWS IoT Jobs service:

- 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 AWS IoT 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 the AWS IoT Jobs service.

#### Job

##### Job Data Type

The `Job` object contains details about a job.

###### Syntax (1)

```
{
    "jobArn": "string",
    "jobId": "string",
    "status": "IN_PROGRESS|CANCELED|SUCCEEDED",
    "forceCanceled": boolean,
    "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,
        "numberOfTimedOutThings": long
    },
    "presignedUrlConfig": {
        "expiresInSec": number,
        "roleArn": "string"
    },
    "jobExecutionsRolloutConfig": {
        "exponentialRate": {
            "baseRatePerMinute": integer,
            "incrementFactor": integer,
            "rateIncreaseCriteria": {
                "numberOfNotifiedThings": integer, // Set one or the other
                "numberOfSucceededThings": integer // of these two values.
            },
            "maximumPerMinute": integer
        }
    },
    "abortConfig": {
}
```

```
"criteriaList": [
    {
        "action": "string",
        "failureType": "string",
        "minNumberOfExecutedThings": integer,
        "thresholdPercentage": integer
    }
],
"timeoutConfig": {
    "inProgressTimeoutInMinutes": long
}
}
```

### 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 SUCCEEDED.

**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 seconds since the epoch, when the job was created.

**lastUpdatedAt**

The time, in seconds since the epoch, when the job was last updated.

**completedAt**

The time, in seconds 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.

`numberOfTimedOutThings`

The number of things whose job execution status is `TIMED_OUT`.

`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 the AWS IoT Jobs service 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 [Create Jobs \(p. 385\)](#).

`jobExecutionRolloutConfig`

Optional. Allows you to create a staged rollout of a job.

`maximumPerMinute`

The maximum number of things (devices) to which the job is sent for execution, per minute.

`exponentialRate`

Allows you to create an exponential rate of rollout for a job.

`baseRatePerMinute`

The minimum number of things that are notified of a pending job, per minute, at the start of job rollout. This parameter allows you to define the initial rate of rollout.

`incrementFactor`

The exponential factor to increase the rate of rollout for a job.

`rateIncreaseCriteria`

The criteria to initiate the increase in rate of rollout for a job. You can specify either `numberOfNotifiedThings` or `numberOfSucceededThing`, but not both.

`numberOfNotifiedThings`

The threshold for number of notified things that initiate the increase in rate of rollout.

`numberOfSucceededThings`

The threshold for number of succeeded things that initiate the increase in rate of rollout.

`abortConfig`

Optional. Details of abort criteria to abort the job.

`criteriaList`

The list of abort criteria to define rules to abort the job.

`action`

The type of abort action to initiate a job abort.

`failureType`

The type of job execution failure to define a rule to initiate a job abort.

`minNumberOfExecutedThings`

Minimum number of executed things before evaluating an abort rule.

`thresholdPercentage`

The threshold as a percentage of the total number of executed things that initiate a job abort.

`timeoutConfig`

Optional. Specifies the amount of time each device has to finish its execution of the job. The timer is started when the job execution status is set to `IN_PROGRESS`. If the job execution status is not set to another terminal state before the time expires, it is set to `TIMED_OUT`.

`inProgressTimeoutInMinutes`

Specifies the amount of time, in minutes, this device has to finish execution of this job. A timer is started, or restarted, whenever this job's execution status is specified as `IN_PROGRESS` with this field populated. If the job execution status is not set to a terminal state before the timer expires, or before another job execution status update is sent with this field populated, the status is set to `TIMED_OUT`.

## JobSummary

### JobSummary Data Type

The `JobSummary` object contains a job summary.

#### Syntax (2)

```
{  
    "jobArn": "string",  
    "jobId": "string",  
    "status": "IN_PROGRESS|CANCELED|SUCCEEDED",  
    "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 SUCCEEDED.

`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

### JobExecution Data Type

The `JobExecution` object represents the execution of a job on a device.

#### Syntax (3)

```
{
    "approximateSecondsBeforeTimedOut": 50,
    "executionNumber": 1234567890,
    "forceCanceled": true|false,
    "jobId": "string",
    "lastUpdatedAt": timestamp,
    "queuedAt": timestamp,
    "startedAt": timestamp,
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|REMOVED",
    "forceCanceled": boolean,
    "statusDetails": {
        "detailsMap": {
            "string": "string" ...
        },
        "status": "string"
    },
    "thingArn": "string",
    "versionNumber": 123
}
```

### Description (3)

`approximateSecondsBeforeTimedOut`

The estimated number of seconds that remain before the job execution status is changed to `TIMED_OUT`. The timeout interval can be anywhere between 1 minute and 7 days (1 to 10080 minutes). The actual job execution timeout can occur up to 60 seconds later than the estimated duration.

`jobId`

The unique identifier you assigned to this job when it was created.

`executionNumber`

A number that identifies this job execution on this 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 seconds since the epoch, when the job execution was queued.

`lastUpdatedAt`

The time, in seconds since the epoch, when the job execution was last updated.

`startedAt`

The time, in seconds since the epoch, when the job execution was started.

`status`

The status of the job execution. Can be one of `QUEUED`, `IN_PROGRESS`, `FAILED`, `SUCCEEDED`, `CANCELED`, `TIMED_OUT`, `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|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|REMOVED"  
}
```

### Description (4)

`executionNumber`

A number that identifies a job execution on a device. It can be used later in commands that return or update job execution information.

`queuedAt`

The time, in seconds since the epoch, when the job execution was queued.

`lastUpdatedAt`

The time, in seconds since the epoch, when the job execution was last updated.

`startAt`

The time, in seconds since the epoch, when the job execution was started.

`status`

The status of the job execution: QUEUED, IN\_PROGRESS, FAILED, SUCCEEDED, CANCELED, TIMED\_OUT, REJECTED, or REMOVED.

## JobExecutionSummaryForJob

### JobExecutionSummaryForJob Data Type

The `JobExecutionSummaryForJob` object contains a summary of information about job executions for a specific job.

#### Syntax (5)

```
{
    "executionSummaries": [
        {
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyThing",
            "jobExecutionSummary": {
                "status": "IN_PROGRESS",
                "lastUpdatedAt": 1549395301.389,
                "queuedAt": 1541526002.609,
                "executionNumber": 1
            }
        },
        ...
    ]
}
```

#### Description (5)

`thingArn`

The AWS IoT thing ARN.

`jobExecutionSummary`

An [JobExecutionSummary \(p. 410\)](#) object.

## JobExecutionSummaryForThing

### JobExecutionSummaryForThing Data Type

The `JobExecutionSummaryForThing` object contains a summary of information about a job execution on a specific thing.

#### Syntax (6)

```
{
    "executionSummaries": [
        {
            "jobExecutionSummary": {
                "status": "IN_PROGRESS",

```

```
        "lastUpdatedAt": 1549395301.389,  
        "queuedAt": 1541526002.609,  
        "executionNumber": 1  
    },  
    "jobId": "MyThingJob"  
},  
...  
]
```

#### Description (6)

**jobId**

The unique identifier you assigned to this job when it was created.

**jobExecutionSummary**

A [JobExecutionSummary \(p. 410\)](#) 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 \(p. 420\)](#). 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.

**CLI (1)**

**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<br><br>member: TargetArn                                    | A list of thing group ARNs that define the targets of the job.                 |
| TargetArn | string                                                           |                                                                                |
| jobId     | string<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+ | The unique identifier you assigned to this job when it was created.            |
| comment   | string<br><br>length max:2028<br><br>pattern: [^\p{C}]+          | An optional string that describes 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<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+ | The unique identifier you assigned to this job when it was created. |
| description | string<br><br>length max:2028<br><br>pattern: [^\p{C}]+          | A short text description of the job.                                |

#### MQTT (1)

Not available.

## CancelJob

### CancelJob Command

Cancels a job.

#### HTTPS (2)

Request:

```
PUT /jobs/jobId/cancel

{
  "force": boolean,
  "comment": "string",
  "reasonCode": "string"
}
```

**jobId**

The unique identifier you assigned to this job when it was created.

**force**

[Optional] If true, job executions with status IN\_PROGRESS and QUEUED are canceled. Otherwise, only job executions with status QUEUED are canceled. The default is false.

#### Warning

Canceling a job with a status of IN\_PROGRESS causes a device that is executing the job to be unable to update the job execution status. Use caution and make sure that each device executing a job that is canceled is able to recover to a valid state.

**comment**

[Optional] A comment string that describes why the job was canceled.

**reasonCode**

[Optional] A reason code string that explains why the job was canceled. If a job is cancelled because it meets the conditions defined by an `abortConfig`, this field is auto-populated.

**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.

**CLI (2)**

**Synopsis:**

```
aws iot cancel-job \
  --job-id <value> \
  [--force <value>] \
  [--comment <value>] \
  [--reasonCode <value>] \
  [--cli-input-json <value>] \
  [--generate-cli-skeleton]
```

**cli-input-json format:**

```
{
    "jobId": "string",
    "force": boolean,
    "comment": "string"
}
```

**cli-input-json fields:**

| Name         | Type                                                             | Description                                                         |
|--------------|------------------------------------------------------------------|---------------------------------------------------------------------|
| <b>jobId</b> | string<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+ | The unique identifier you assigned to this job when it was created. |
| <b>force</b> | boolean                                                          | If true, jobs with status QUEUED and IN_PROGRESS                    |

| Name       | Type                                                        | Description                                                                                                                                                                                                                                                                                                                                                   |
|------------|-------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |                                                             | <p>are canceled. Otherwise, only jobs with status QUEUED are canceled.</p> <p><b>Warning</b><br/>Canceling a job with a status of IN_PROGRESS causes a device that is executing the job to be unable to update the job execution status. Use caution and make sure that each device executing a job that is canceled is able to recover to a valid state.</p> |
| comment    | string<br>length max:2028<br>pattern: [^\p{C}]+             | An optional string that describes why the job was canceled.                                                                                                                                                                                                                                                                                                   |
| reasonCode | string<br>length max:128<br>pattern: [\p{Upper}\p{Digit}_]+ | An optional string that explains why the job was canceled. If the job is canceled because it meets the criteria defined by the abortConfig, this field is auto-populated.                                                                                                                                                                                     |

Output:

```
{
  "jobArn": "string",
  "jobId": "string",
  "description": "string"
}
```

#### CLI output fields:

| Name        | Type                                                     | Description                                                         |
|-------------|----------------------------------------------------------|---------------------------------------------------------------------|
| jobArn      | string                                                   | The job ARN.                                                        |
| jobId       | string<br>length max:64 min:1<br>pattern: [a-zA-Z0-9_-]+ | The unique identifier you assigned to this job when it was created. |
| description | string<br>length max:2028<br>pattern: [^\p{C}]+          | A short text description of the job.                                |

## MQTT (2)

Not available.

## CancelJobExecution

### CancelJobExecution Command

Cancels a job execution on a device.

### HTTPS (3)

Request:

```
PUT /things/thingName/jobs/jobId/cancel

{
    "force": boolean,
    "expectedVersion": "string",
    "statusDetails": {
        "string": "string"
        ...
    }
}
```

*thingName*

The name of the thing whose job execution will be canceled.

*jobId*

The unique identifier you assigned to the job when it was created.

*force*

Optional. If true, a job execution with a status of IN\_PROGRESS or QUEUED can be canceled. Otherwise, only a job execution with status of QUEUED can be canceled. If you attempt to cancel a job execution with a status of IN\_PROGRESS and you do not set force to true, an InvalidStateException is thrown. The default is false.

#### Warning

Canceling a job with a status of IN\_PROGRESS causes a device that is executing the job to be unable to update the job execution status. Use caution and make sure that each device executing a job that is canceled is able to recover to a valid state.

*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 the AWS IoT Jobs service does not match, the update is rejected with a VersionConflictException error, and an ErrorResponse that contains the current job execution status data is returned. (This makes it unnecessary to perform a separate DescribeJobExecution request to obtain the job execution status data.)

*statusDetails*

Optional. A collection of name-value pairs that describe the status of the job execution.

Response:

```
{
```

}

CLI (3)

**Synopsis:**

```
aws iot cancel-job-execution \
    --job-id <value> \
    --thing-name <value> \
    [--force | --no-force] \
    [--expected-version <value>] \
    [--status-details <value>] \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

**cli-input-json format:**

```
{
  "jobId": "string",
  "thingName": "string",
  "force": boolean,
  "expectedVersion": long,
  "statusDetails": {
    "string": "string"
  }
}
```

**cli-input-json fields:**

| Name      | Type                                                              | Description                                                                                                                                                                                                                                                                                                                                                       |
|-----------|-------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| jobId     | string<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+  | The job to be canceled.                                                                                                                                                                                                                                                                                                                                           |
| thingName | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9_-]+ | The name of the thing whose execution of the job will be canceled.                                                                                                                                                                                                                                                                                                |
| force     | boolean                                                           | Optional. If true, the job execution is canceled if it has status of IN_PROGRESS or QUEUED. Otherwise, the job execution is canceled only if it has status of QUEUED. However, if you attempt to cancel a job execution that has a status of IN_PROGRESS, and you do not set --force to true, an InvalidStateTransitionException is thrown. The default is false. |

**Warning**

Cancelling a job that has a status

| Name            | Type                                                              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------|-------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 |                                                                   | of IN_PROGRESS, causes a device that is executing the job to be unable to update the job execution status. Use caution and make sure that each device executing a job that is canceled is able to recover to a valid state.                                                                                                                                                                                                                                                                                                       |
| expectedVersion | long<br><br>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 the AWS IoT Jobs service does not match, the update is rejected with a <code>VersionMismatch</code> error, and an <code>ErrorResponse</code> that contains the current job execution status data is returned. (This makes it unnecessary to perform a separate <code>DescribeJobExecution</code> request to obtain the job execution status data.) |
| statusDetails   | map<br><br>key: DetailsKey<br><br>value: DetailsValue             | A collection of name-value pairs that describe the status of the job execution. If not specified, the <code>statusDetails</code> are unchanged.                                                                                                                                                                                                                                                                                                                                                                                   |
| DetailsKey      | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9:_]+ |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| DetailsValue    | string<br><br>length max:1024 min:1<br><br>pattern: [^\p{C}]*+    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

Output:

None

MQTT (3)

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.

A job can have an optional `TimeoutConfig`, which sets the value of the in-progress timer. The in-progress timer can't be updated and applies to all executions of 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 in your AWS account.

### HTTPS (4)

Request:

```
PUT /jobs/jobId
{
    "targets": [ "string" ],
    "document": "string",
    "documentSource": "string",
    "description": "string",
    "presignedUrlConfigData": {
        "roleArn": "string",
        "expiresInSec": "integer"
    },
    "targetSelection": "CONTINUOUS|SNAPSHOT",
    "jobExecutionsRolloutConfig": {
        "exponentialRate": {
            "baseRatePerMinute": integer,
            "incrementFactor": integer,
            "rateIncreaseCriteria": {
                "numberOfNotifiedThings": integer, // Set one or the other
                "numberOfSucceededThings": integer // of these two values.
            },
            "maximumPerMinute": integer
        }
    },
    "abortConfig": {
        "criteriaList": [
            {
                "action": "string",
                "failureType": "string",
                "minNumberOfExecutedThings": integer,
            }
        ]
    }
}
```

```
        "thresholdPercentage": integer
    }
],
},
"timeoutConfig": {
    "inProgressTimeoutInMinutes": long
}
}
```

**jobId**

A job identifier, which must be unique for your AWS account. We recommend using a UUID. Alphanumeric 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 [Create Jobs \(p. 385\)](#).

**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 the AWS IoT Jobs service 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 is not more than the specified value.

**exponentialRate**

Allows you to create an exponential rate of rollout for a job.

`baseRatePerMinute`

The minimum number of things that are notified of a pending job, per minute, at the start of job rollout. This parameter allows you to define the initial rate of rollout.

`incrementFactor`

The exponential factor to increase the rate of rollout for a job.

`rateIncreaseCriteria`

The criteria to initiate the increase in rate of rollout for a job. Set values for either the `numberOfNotifiedThings` or `numberOfSucceededThings`, but not both.

`numberOfNotifiedThings`

The threshold for number of notified things that initiate the increase in rate of rollout.

`numberOfSucceededThings`

The threshold for number of succeeded things that initiate the increase in rate of rollout.

`abortConfig`

Optional. Details of abort criteria to abort the job.

`criteriaList`

The list of abort criteria to define rules to abort the job.

`action`

The type of abort action to initiate a job abort.

`failureType`

The type of job execution failure to define a rule to initiate a job abort.

`minNumberOfExecutedThings`

Minimum number of executed things before evaluating an abort rule.

`thresholdPercentage`

The threshold as a percentage of the total number of executed things that initiate a job abort.

`timeoutConfig`

Optional. Specifies the amount of time each device has to finish its execution of the job. The timer is started when the job execution status is set to `IN_PROGRESS`. If the job execution status is not set to another terminal state before the time expires, it is set to `TIMED_OUT`.

`inProgressTimeoutInMinutes`

Specifies the amount of time, in minutes, this device has to finish execution of this job. A timer is started, or restarted, whenever this job's execution status is specified as `IN_PROGRESS` with this field populated. If the job execution status is not set to a terminal state before the timer expires, or before another job execution status update is sent with this field populated, the status is set to `TIMED_OUT`.

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.

**CLI (4)**

**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>] \
[--abort-config <value>] \
[--timeout-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": {
    "exponentialRate": {
      "baseRatePerMinute": integer,
      "incrementFactor": integer,
      "rateIncreaseCriteria": {
        "numberOfNotifiedThings": integer, // Set one or the other
        "numberOfSucceededThings": integer // of these two values.
      },
      "maximumPerMinute": integer
    }
  },
  "abortConfig": {
    "criteriaList": [
      {
        "action": "string",
        "condition": "string"
      }
    ]
  }
}
```

```

        "failureType": "string",
        "minNumberOfExecutedThings": integer,
        "thresholdPercentage": integer
    }
]
},
"timeoutConfig": {
    "inProgressTimeoutInMinutes": long
},
"documentParameters": {
    "string": "string"
}
}

```

**cli-input-json fields:**

| Name               | Type                                                                 | Description                                                                                                                                                                                                |
|--------------------|----------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| jobId              | string<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+     | A job identifier which must be unique for your AWS account. We recommend using a UUID. Alphanumeric characters, "-" and "_" are valid for use here.                                                        |
| targets            | list<br><br>member: TargetArn                                        | A list of things and thing groups to which the job should be sent.                                                                                                                                         |
| TargetArn          | string                                                               |                                                                                                                                                                                                            |
| documentSource     | string<br><br>length max:1350 min:1                                  | An S3 link to the job document.                                                                                                                                                                            |
| document           | string<br><br>length max:32768                                       | The job document.                                                                                                                                                                                          |
| description        | string<br><br>length max:2028<br><br>pattern: [^\p{C}]+              | A short text description of the job.                                                                                                                                                                       |
| presignedUrlConfig | PresignedUrlConfig                                                   | Configuration information for presigned S3 URLs.                                                                                                                                                           |
| roleArn            | string<br><br>length max:2048 min:20                                 | The ARN of an IAM role that grants permission to download files from the Amazon S3 bucket where the job data or updates are stored. The role must also grant permission for AWS IoT to download the files. |
| expiresInSec       | long<br><br>java class: java.lang.Long<br><br>range- max:3600 min:60 | How long (in seconds) presigned URLs are valid. Valid values are 60 - 3600. The default value is 3600 seconds. Presigned URLs are generated when the AWS IoT Jobs service                                  |

| Name                       | Type                                                                      | Description                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------------|---------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                            |                                                                           | receives an MQTT request for the job document.                                                                                                                                                                                                                                                                                                                                                        |
| targetSelection            | string<br><br>enum: CONTINUOUS   SNAPSHOT                                 | 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 can 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. |
| jobExecutionsRolloutConfig | JobExecutionsRolloutConfig                                                | Allows you to create a staged rollout of the job.                                                                                                                                                                                                                                                                                                                                                     |
| maximumPerMinute           | integer<br><br>java class: java.lang.Integer<br><br>range- max:1000 min:1 | The maximum number of things that are notified of a pending job, per minute. This parameter allows you to create a staged rollout.                                                                                                                                                                                                                                                                    |
| exponentialRate            | ExponentialRolloutRate                                                    | The rate of increase for a job rollout. This parameter allows you to define an exponential rate for a job rollout.                                                                                                                                                                                                                                                                                    |
| baseRatePerMinute          | java class: java.lang.Integer                                             | The minimum number of things that will be notified of a pending job, per minute at the start of job rollout. This parameter allows you to define the initial rate of rollout.                                                                                                                                                                                                                         |
| incrementFactor            | java class: java.lang.Double                                              | The exponential factor to increase the rate of rollout for a job.                                                                                                                                                                                                                                                                                                                                     |
| rateIncreaseCriteria       | RateIncreaseCriteria                                                      | Allows you to define a criteria to initiate the increase in rate of rollout for a job. Set a value for either <code>numberOfNotifiedThings</code> or <code>numberOfSucceededThings</code> , but not both.                                                                                                                                                                                             |
| numberOfNotifiedThings     | java class: java.lang.Double                                              | The threshold for number of notified things that will initiate the increase in rate of rollout.                                                                                                                                                                                                                                                                                                       |

| Name                      | Type                                                               | Description                                                                                                                                                                                                                                                                  |
|---------------------------|--------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| numberOfSucceededThings   | java class: java.lang.Double                                       | The threshold for number of succeeded things that will initiate the increase in rate of rollout.                                                                                                                                                                             |
| abortConfig               | AbortConfig                                                        | Allows you to create criteria to abort a job.                                                                                                                                                                                                                                |
| criteriaList              | AbortCriteria                                                      | The list of abort criteria to define rules to abort the job.                                                                                                                                                                                                                 |
| action                    | java class: java.lang.String (CANCEL)                              | The type of abort action to initiate a job abort.                                                                                                                                                                                                                            |
| failureType               | java class: java.lang.String (FAILED   REJECTED   TIMED_OUT   ALL) | The type of job execution failure to define a rule to initiate a job abort.                                                                                                                                                                                                  |
| minNumberOfExecutedThings | java class: java.lang.Integer                                      | Minimum number of executed things before evaluating an abort rule.                                                                                                                                                                                                           |
| thresholdPercentage       | java class: java.lang.Double                                       | <p>The threshold as a percentage of the total number of executed things that will initiate a job abort.</p> <p>AWS IoT supports up to two digits after the decimal (for example, 10.9 and 10.99, but not 10.999).</p>                                                        |
| timeoutConfig             | TimeoutConfig                                                      | Specifies the amount of time each device has to finish its execution of the job. The timer is started when the job execution status is set to IN_PROGRESS. If the job execution status is not set to another terminal state before the time expires, it is set to TIMED_OUT. |

| Name                       | Type                                                               | Description                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------------|--------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| inProgressTimeoutInMinutes | long                                                               | Specifies the amount of time, in minutes, this device has to finish execution of this job. A timer is started, or restarted, whenever this job's execution status is specified as IN_PROGRESS with this field populated. If the job execution status is not set to a terminal state before the timer expires, or before another job execution status update is sent with this field populated, the status is set to TIMED_OUT. |
| documentParameters         | map<br>key: ParameterKey<br>value: ParameterValue                  | Parameters for the job document.                                                                                                                                                                                                                                                                                                                                                                                               |
| ParameterKey               | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9:_-]+ |                                                                                                                                                                                                                                                                                                                                                                                                                                |
| ParameterValue             | string<br><br>length max:1024 min:1<br><br>pattern: [^\p{C}]+      |                                                                                                                                                                                                                                                                                                                                                                                                                                |

Output:

```
{
  "jobArn": "string",
  "jobId": "string",
  "description": "string"
}
```

#### CLI output fields:

| Name        | Type                                                             | Description                                     |
|-------------|------------------------------------------------------------------|-------------------------------------------------|
| jobArn      | string                                                           | The job ARN.                                    |
| jobId       | string<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+ | The unique identifier you assigned to this job. |
| description | string<br><br>length max:2028                                    | The job description.                            |

| Name | Type                | Description |
|------|---------------------|-------------|
|      | pattern: [^\\p{C}]+ |             |

## MQTT (4)

Not available.

## DeleteJob

### DeleteJob Command

Deletes a job and its related job executions.

Deleting a job can take time, depending on the number of job executions created for the job and various other factors. While the job is being deleted, the status of the job is shown as "DELETION\_IN\_PROGRESS". Attempting to delete or cancel a job whose status is already "DELETION\_IN\_PROGRESS" results in an error.

## HTTPS (5)

### Request syntax:

```
DELETE /jobs/jobId?force=force
```

### URI Request Parameters:

| Name  | Type      | Req? | Description                                                                                                                                                                                                           |
|-------|-----------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| jobId | JobId     | yes  | The ID of the job to be deleted.                                                                                                                                                                                      |
| force | ForceFlag | no   | (Optional) When true, you can delete a job with a status of "IN_PROGRESS". Otherwise, you can only delete a job that is in a terminal state ("SUCCEEDED" or "CANCELED") or an exception occurs. The default is false. |

**Note**  
Deleting a job with a status of "IN\_PROGRESS", causes a device that is executing the job to be unable to access job information or update the job execution

| Name | Type | Req? | Description                                                                                                             |
|------|------|------|-------------------------------------------------------------------------------------------------------------------------|
|      |      |      | status. Use caution and make sure that each device executing a job that is deleted is able to recover to a valid state. |

**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

**InvalidStateTransitionException**

An update attempted to change the job or job execution to a state that is invalid because of its current state (for example, an attempt to change a request in state SUCCEEDED to state IN\_PROGRESS). In this case, the body of the error message also contains the executionState field.

HTTP response code: 409

**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 (5)**

**Synopsis:**

```
aws iot delete-job \
--job-id <value> \
[--force | --no-force] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "jobId": "string",
  "force": boolean
}
```

**cli-input-json fields:**

| Name  | Type                                                             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------|------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| jobId | string<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+ | The ID of the job to be deleted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| force | boolean                                                          | (Optional) When true, you can delete a job with a status of IN_PROGRESS. Otherwise, you can only delete a job that is in a terminal state (SUCCEEDED or CANCELED) or an exception occurs. The default is false.<br><br><b>Note</b><br>Deleting a job with a status of IN_PROGRESS, causes a device that is executing the job to be unable to access job information or update the job execution status. Use caution and make sure that each device executing a job that is deleted is able to recover to a valid state. |

**Output:**

None

MQTT (5)

Not available.

## DeleteJobExecution

### DeleteJobExecution Command

Deletes a job execution.

HTTPS (6)

**Request syntax:**

```
DELETE /things/thingName/jobs/jobId/executionNumber/executionNumber?force=force
```

**URI Request Parameters:**

| Name            | Type            | Req? | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------|-----------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| jobId           | JobId           | yes  | The ID of the job whose execution will be deleted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| thingName       | ThingName       | yes  | The name of the thing whose execution of the job will be deleted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| executionNumber | ExecutionNumber | yes  | The ID of the job execution to be deleted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| force           | ForceFlag       | no   | <p>When true, you can delete a job execution with a status of IN_PROGRESS. Otherwise, you can only delete a job execution that is in a terminal state (SUCCEEDED, FAILED, TIMED_OUT, REJECTED, REMOVED, or CANCELED) or an exception occurs. The default is false.</p> <p><b>Note</b><br/>           Deleting a job execution with a status of IN_PROGRESS causes the device to be unable to access job information or update the job execution status. Use caution and make sure that the device is able to recover to a valid state.</p> |

**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

#### `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 `SUCCEEDED` to state `IN_PROGRESS`). In this case, the body of the error message also contains the `executionState` field.

HTTP response code: 409

#### `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 (6)

#### **Synopsis:**

```
aws iot delete-job-execution \
  --job-id <value> \
  --thing-name <value> \
  --execution-number <value> \
  [--force | --no-force] \
  [--cli-input-json <value>] \
  [--generate-cli-skeleton]
```

**cli-input-json format:**

```
{
  "jobId": "string",
  "thingName": "string",
  "executionNumber": long,
  "force": boolean
}
```

#### **cli-input-json fields:**

| Name  | Type                              | Description                                        |
|-------|-----------------------------------|----------------------------------------------------|
| jobId | string<br><br>length max:64 min:1 | The ID of the job whose execution will be deleted. |

| Name            | Type                                                               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------------|--------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 | pattern: [a-zA-Z0-9_-]+                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| thingName       | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9:_-]+ | The name of the thing whose execution of the job will be deleted.                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| executionNumber | long<br><br>java class: java.lang.Long                             | The ID of the job execution to be deleted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| force           | boolean                                                            | When true, you can delete a job execution with a status of IN_PROGRESS. Otherwise, you can only delete a job execution that is in a terminal state (SUCCEEDED, FAILED, TIMED_OUT, REJECTED, REMOVED, or CANCELED) or an exception occurs. The default is false.<br><br><b>Note</b><br>Deleting a job execution with a status of IN_PROGRESS causes the device to be unable to access job information or update the job execution status. Use caution and make sure that the device is able to recover to a valid state. |

Output:

None

MQTT (6)

Not available.

## DescribeJob

DescribeJob Command

Gets the details of the specified job.

HTTPS (7)

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 \(p. 405\)](#) object.

CLI (7)

**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                                                         |
|--------------------|------------------------------------------------------------------|---------------------------------------------------------------------|
| <code>jobId</code> | string<br><br>length max:64 min:1<br><br>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",  
        "forceCanceled": boolean,  
        "comment": "string",  
        "targets": [  
            "string"  
        ],  
        "description": "string",  
    },  
}
```

```

"presignedUrlConfig": {
    "roleArn": "string",
    "expiresInSec": long
},
"jobExecutionsRolloutConfig": {
    "exponentialRate": {
        "baseRatePerMinute": integer,
        "incrementFactor": integer,
        "rateIncreaseCriteria": {
            "numberOfNotifiedThings": integer, // Set one or the other
            "numberOfSucceededThings": integer // of these two values.
        },
        "maximumPerMinute": integer
    }
},
"abortConfig": {
    "criteriaList": [
        {
            "action": "string",
            "failureType": "string",
            "minNumberOfExecutedThings": integer,
            "thresholdPercentage": integer
        }
    ]
},
"createdAt": "timestamp",
"lastUpdatedAt": "timestamp",
"completedAt": "timestamp",
"jobProcessDetails": {
    "processingTargets": [
        "string"
    ],
    "numberOfCanceledThings": "integer",
    "numberOfSucceededThings": "integer",
    "numberOfFailedThings": "integer",
    "numberOfRejectedThings": "integer",
    "numberOfQueuedThings": "integer",
    "numberOfInProgressThings": "integer",
    "numberOfRemovedThings": "integer",
    "numberOfTimedOutThings": "integer"
},
"documentParameters": {
    "string": "string"
},
"timeoutConfig": {
    "inProgressTimeoutInMinutes": number
}
}
}
}

```

#### CLI output fields:

| Name           | Type                                | Description                                                                        |
|----------------|-------------------------------------|------------------------------------------------------------------------------------|
| documentSource | string<br><br>length max:1350 min:1 | An Amazon S3 link to the job document.                                             |
| job            | Job                                 | Information about the job.                                                         |
| jobArn         | string                              | An ARN that identifies the job with format "arn:aws:iot:region:account:job/jobId". |

| Name               | Type                                                             | Description                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------|------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| jobId              | string<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+ | The unique identifier you assigned to this job when it was created.                                                                                                                                                                                                                                                                                                                                                            |
| targetSelection    | string<br><br>enum: CONTINUOUS   SNAPSHOT                        | 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 can also be run on a thing when a change is detected in a target. For example, a job runs 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<br><br>enum: IN_PROGRESS   CANCELED   SUCCEEDED           | The status of the job, one of IN_PROGRESS, CANCELED, or SUCCEEDED.                                                                                                                                                                                                                                                                                                                                                             |
| forceCanceled      | boolean<br><br>java class: java.lang.Boolean                     | Is true if the job was canceled with the optional force parameter set to true.                                                                                                                                                                                                                                                                                                                                                 |
| comment            | string<br><br>length max:2028<br><br>pattern: [^\p{C}]+          | If the job was updated, describes the reason for the update.                                                                                                                                                                                                                                                                                                                                                                   |
| targets            | list<br><br>member: TargetArn                                    | A list of AWS IoT things and thing groups to which the job should be sent.                                                                                                                                                                                                                                                                                                                                                     |
| TargetArn          | string                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                |
| description        | string<br><br>length max:2028<br><br>pattern: [^\p{C}]+          | A short text description of the job.                                                                                                                                                                                                                                                                                                                                                                                           |
| presignedUrlConfig | PresignedUrlConfig                                               | Configuration for presigned Amazon S3 URLs.                                                                                                                                                                                                                                                                                                                                                                                    |

| Name                       | Type                                                                      | Description                                                                                                                                                                                                              |
|----------------------------|---------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| roleArn                    | string<br><br>length max:2048 min:20                                      | The ARN of an IAM role that grants permission to download files from the Amazon S3 bucket where the job data or updates are stored. The role must also grant permission for tAWS IoT Jobs service to download the files. |
| expiresInSec               | long<br><br>java class: java.lang.Long<br><br>range- max:3600 min:60      | How long (in seconds) presigned URLs are valid. Valid values are 60 - 3600. The default value is 3600 seconds. Presigned URLs are generated when the AWS IoT Jobs service receives an MQTT request for the job document. |
| jobExecutionsRolloutConfig | JobExecutionsRolloutConfig                                                | Allows you to create a staged rollout of the job.                                                                                                                                                                        |
| maximumPerMinute           | integer<br><br>java class: java.lang.Integer<br><br>range- max:1000 min:1 | The maximum number of things that are notified of a pending job, per minute. This parameter allows you to create a staged rollout.                                                                                       |
| exponentialRate            | ExponentialRolloutRate                                                    | The rate of increase for a job rollout. This parameter allows you to define an exponential rate for a job rollout.                                                                                                       |
| baseRatePerMinute          | java class: java.lang.Integer                                             | The minimum number of things that are notified of a pending job, per minute at the start of job rollout. This parameter allows you to define the initial rate of rollout.                                                |
| incrementFactor            | java class: java.lang.Double                                              | The exponential factor to increase the rate of rollout for a job.                                                                                                                                                        |
| rateIncreaseCriteria       | RateIncreaseCriteria                                                      | Allows you to define a criteria to initiate the increase in rate of rollout for a job. Set a value for either <code>numberOfNotifiedThings</code> or <code>numberOfSucceededThings</code> , but not both.                |
| numberOfNotifiedThings     | java class: java.lang.Double                                              | The threshold for number of notified things that initiate the increase in rate of rollout.                                                                                                                               |

| Name                      | Type                                                                  | Description                                                                                                                                                                                                      |
|---------------------------|-----------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| numberOfSucceededThings   | java class: java.lang.Double                                          | The threshold for number of succeeded things that initiate the increase in rate of rollout.                                                                                                                      |
| abortConfig               | AbortConfig                                                           | Allows you to create criteria to abort a job.                                                                                                                                                                    |
| criteriaList              | AbortCriteria                                                         | The list of abort criteria to define rules to abort the job.                                                                                                                                                     |
| action                    | java class: java.lang.String (CANCEL)                                 | The type of abort action to initiate a job abort.                                                                                                                                                                |
| failureType               | java class: java.lang.String (FAILED   REJECTED   TIMED_OUT   ALL)    | The type of job execution failure to define a rule to initiate a job abort.                                                                                                                                      |
| minNumberOfExecutedThings | java class: java.lang.Integer                                         | Minimum number of executed things before evaluating an abort rule.                                                                                                                                               |
| thresholdPercentage       | java class: java.lang.Double                                          | <p>The threshold as a percentage of the total number of executed things that initiate a job abort.</p> <p>AWS IoT supports up to two digits after the decimal (for example, 10.9 and 10.99, but not 10.999).</p> |
| createdAt                 | timestamp                                                             | The time, in seconds since the epoch, when the job was created.                                                                                                                                                  |
| lastUpdatedAt             | timestamp                                                             | The time, in seconds since the epoch, when the job was last updated.                                                                                                                                             |
| completedAt               | timestamp                                                             | The time, in seconds since the epoch, when the job was completed.                                                                                                                                                |
| jobProcessDetails         | JobProcessDetails                                                     | Details about the job process.                                                                                                                                                                                   |
| processingTargets         | list<br>member:<br>ProcessingTargetName<br>java class: java.util.List | The devices on which the job is executing.                                                                                                                                                                       |
| ProcessingTargetName      | string                                                                |                                                                                                                                                                                                                  |
| numberOfCanceledThings    | integer<br>java class: java.lang.Integer                              | The number of things that canceled the job.                                                                                                                                                                      |

| Name                     | Type                                                               | Description                                                                                                                                                                                                                                                                                 |
|--------------------------|--------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| numberOfSucceededThings  | integer<br>java class: java.lang.Integer                           | The number of things that successfully completed the job.                                                                                                                                                                                                                                   |
| numberOfFailedThings     | integer<br>java class: java.lang.Integer                           | The number of things that failed executing the job.                                                                                                                                                                                                                                         |
| numberOfRejectedThings   | integer<br>java class: java.lang.Integer                           | The number of things that rejected the job.                                                                                                                                                                                                                                                 |
| numberOfQueuedThings     | integer<br>java class: java.lang.Integer                           | The number of things that are awaiting execution of the job.                                                                                                                                                                                                                                |
| numberOfInProgressThings | integer<br>java class: java.lang.Integer                           | The number of things currently executing the job.                                                                                                                                                                                                                                           |
| numberOfRemovedThings    | integer<br>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.                                                                                                                       |
| numberOfTimedOutThings   | integer<br>java class: java.lang.Integer                           | The number of things whose job execution status is <b>TIMED_OUT</b> .                                                                                                                                                                                                                       |
| documentParameters       | map<br>key: ParameterKey<br>value: ParameterValue                  | The parameters specified for the job document.                                                                                                                                                                                                                                              |
| ParameterKey             | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9:_-]+ |                                                                                                                                                                                                                                                                                             |
| ParameterValue           | string<br><br>length max:1024 min:1<br><br>pattern: [^\p{C}]+      |                                                                                                                                                                                                                                                                                             |
| timeoutConfig            | TimeoutConfig                                                      | Specifies the amount of time each device has to finish its execution of the job. A timer is started when the job execution status is set to <b>IN_PROGRESS</b> . If the job execution status is not set to another terminal state before the timer expires, it is set to <b>TIMED_OUT</b> . |

| Name                       | Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| inProgressTimeoutInMinutes | long | Specifies the amount of time, in minutes, this device has to finish execution of this job. The timeout interval can be anywhere between 1 minute and 7 days (1 to 10080 minutes). The in-progress timer can't be updated and applies to all job executions for the job. Whenever a job execution remains in the IN_PROGRESS status for longer than this interval, the job execution fails and switches to the terminal TIMED_OUT status. |

## MQTT (7)

Not available.

## DescribeJobExecution

### DescribeJobExecution Command

Gets details of a job execution. The job's execution status must be SUCCEEDED or FAILED.

### HTTPS (8)

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 job execution on a device. (See [JobExecution \(p. 409\)](#).) If not specified, the latest job execution is returned.

Response:

```
{
  "execution": { JobExecution }
}
```

**execution**

A [JobExecution \(p. 409\)](#) object.

## CLI (8)

### 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<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+  | The unique identifier you assigned to this job when it was created.                                                            |
| thingName       | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9_-]+ | The name of the thing on which the job execution is running.                                                                   |
| executionNumber | long<br><br>java class: java.lang.Long                            | A string (consisting of the digits "0" through "9") that is used to specify a particular job execution on a particular device. |

### Output:

```
{
    "execution": {
        "approximateSecondsBeforeTimedOut": "number"
        "jobId": "string",
        "status": "string",
        "forceCanceled": boolean,
        "statusDetails": {
            "detailsMap": {
                "string": "string"
            }
        },
        "thingArn": "string",
        "queuedAt": "timestamp",
        "startedAt": "timestamp",
        "lastUpdatedAt": "timestamp",
        "executionNumber": long,
```

```
        "versionNumber": long
    }
}
```

**CLI output fields:**

| Name                             | Type                                                                                                      | Description                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------------------|-----------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| execution                        | JobExecution                                                                                              | Information about the job execution.                                                                                                                                                                                                                                                                                                                                                    |
| approximateSecondsBeforeTimedOut | long                                                                                                      | The estimated number of seconds that remain before the job execution status is changed to <code>TIMED_OUT</code> . The timeout interval can be anywhere between 1 minute and 7 days (1 to 10080 minutes). The actual job execution timeout can occur up to 60 seconds later than the estimated duration. This value is not included if the job execution has reached a terminal status. |
| jobId                            | string<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+                                          | The unique identifier you assigned to the job when it was created.                                                                                                                                                                                                                                                                                                                      |
| status                           | string<br><br>enum: QUEUED   IN_PROGRESS   SUCCEEDED   FAILED   TIMED_OUT   REJECTED   REMOVED   CANCELED | The status of the job execution (IN_PROGRESS, QUEUED, FAILED, SUCCEEDED, TIMED_OUT, CANCELED, or REJECTED).                                                                                                                                                                                                                                                                             |
| forceCanceled                    | boolean<br><br>java class: java.lang.Boolean                                                              | Is <code>true</code> if the job execution was canceled with the optional <code>force</code> parameter set to <code>true</code> .                                                                                                                                                                                                                                                        |
| statusDetails                    | JobExecutionStatusDetails                                                                                 | A collection of name-value pairs that describe the status of the job execution.                                                                                                                                                                                                                                                                                                         |
| detailsMap                       | map<br><br>key: DetailsKey<br><br>value: DetailsValue                                                     | The job execution status.                                                                                                                                                                                                                                                                                                                                                               |
| DetailsKey                       | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9_-]+                                         |                                                                                                                                                                                                                                                                                                                                                                                         |
| DetailsValue                     | string                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                         |

| Name            | Type                                         | Description                                                                                                                                                                        |
|-----------------|----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 | length max:1024 min:1<br>pattern: [^\p{C}]*+ |                                                                                                                                                                                    |
| thingArn        | string                                       | The ARN of the thing on which the job execution is running.                                                                                                                        |
| queuedAt        | timestamp                                    | The time, in seconds since the epoch, when the job execution was queued.                                                                                                           |
| startedAt       | timestamp                                    | The time, in seconds since the epoch, when the job execution started.                                                                                                              |
| lastUpdatedAt   | timestamp                                    | The time, in seconds since the epoch, when the job execution was last updated.                                                                                                     |
| executionNumber | long<br>java class: java.lang.Long           | A string (consisting of the digits "0" through "9") that identifies this job execution on this device. It can be used in commands that return or update job execution information. |
| versionNumber   | long                                         | The version of the job execution. Job execution versions are incremented each time they are updated by a device.                                                                   |

## MQTT (8)

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 the AWS IoT Jobs service receives a request over MQTT.

## HTTPS (9)

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.

CLI (9)

**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<br><br>length max:64 min:1<br><br>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<br><br>length max:32768 | The job document content. |

MQTT (9)

Not available.

## ListJobExecutionsForJob

### ListExecutionsForJob Command

Gets a list of job executions for a job.

HTTPS (10)

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, SUCCEEDED, FAILED, TIMED\_OUT, 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 \(p. 410\)](#) objects associated with the specified job ID.

CLI (10)

**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<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+                                          | The unique identifier you assigned to this job when it was created. |
| status     | string<br><br>enum: QUEUED   IN_PROGRESS   SUCCEEDED   FAILED   TIMED_OUT   REJECTED   REMOVED   CANCELED | The status of the job.                                              |
| maxResults | integer<br><br>java class: java.lang.Integer<br><br>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<br><br>member: JobExecutionSummaryForJob<br><br>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. |

| Name                | Type                                                                                                      | Description                                                                                                                                                                              |
|---------------------|-----------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| jobExecutionSummary | JobExecutionSummary                                                                                       | Contains a subset of information about a job execution.                                                                                                                                  |
| status              | string<br><br>enum: QUEUED   IN_PROGRESS   SUCCEEDED   FAILED   TIMED_OUT   REJECTED   REMOVED   CANCELED | The status of the job execution.                                                                                                                                                         |
| queuedAt            | timestamp                                                                                                 | The time, in seconds since the epoch, when the job execution was queued.                                                                                                                 |
| startedAt           | timestamp                                                                                                 | The time, in seconds since the epoch, when the job execution started.                                                                                                                    |
| lastUpdatedAt       | timestamp                                                                                                 | The time, in seconds since the epoch, when the job execution was last updated.                                                                                                           |
| executionNumber     | long<br><br>java class: java.lang.Long                                                                    | A string (consisting of the digits "0" through "9") that identifies this job execution on this device. It can be used later in commands that return or update job execution information. |
| nextToken           | string                                                                                                    | The token for the next set of results, or <b>null</b> if there are no additional results.                                                                                                |

## MQTT (10)

Not available.

## ListJobExecutionsForThing

### ListJobExecutionsForThing Command

Gets a list of job executions for a thing.

### HTTPS (11)

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, SUCCEEDED, FAILED, TIMED\_OUT, 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 \(p. 410\)](#) objects for the job executions associated with the specified thing.

**CLI (11)**

**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<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9:_-]+        | The thing name.                                                                  |
| status    | string<br><br>enum: QUEUED   IN_PROGRESS   SUCCEEDED   FAILED   TIMED_OUT | An optional filter that lets you search for jobs that have the specified status. |

| Name       | Type                                                                     | Description                                               |
|------------|--------------------------------------------------------------------------|-----------------------------------------------------------|
|            | REJECTED   REMOVED   CANCELED                                            |                                                           |
| maxResults | integer<br><br>java class: java.lang.Integer<br><br>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<br><br>member:<br>JobExecutionSummaryForThing<br><br>java class: java.util.List | A list of job execution summaries.                                  |
| JobExecutionSummaryForThing | JobExecutionSummaryForThing                                                          |                                                                     |
| jobId                       | string<br><br>length max:64 min:1<br><br>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<br><br>enum: QUEUED   IN_PROGRESS   SUCCEEDED   FAILED   TIMED_OUT            | The status of the job execution.                                    |

| Name            | Type                                   | Description                                                                                                                                                                              |
|-----------------|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 | REJECTED   REMOVED   CANCELED          |                                                                                                                                                                                          |
| queuedAt        | timestamp                              | The time, in seconds since the epoch, when the job execution was queued.                                                                                                                 |
| startedAt       | timestamp                              | The time, in seconds since the epoch, when the job execution started.                                                                                                                    |
| lastUpdatedAt   | timestamp                              | The time, in seconds since the epoch, when the job execution was last updated.                                                                                                           |
| executionNumber | long<br><br>java class: java.lang.Long | A string (consisting of the digits "0" through "9") that identifies this job execution on this device. It can be used later in commands that return or update job execution information. |
| nextToken       | string                                 | The token for the next set of results, or <b>null</b> if there are no additional results.                                                                                                |

## MQTT (11)

Not available.

## ListJobs

### ListJobs Command

Gets a list of the jobs in your AWS account.

### HTTPS (12)

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 SUCCEEDED.

**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 \(p. 408\)](#) objects, one for each job in your AWS account that matches the specified filtering criteria.

**CLI (12)**

**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                                                                                              |
|------------------------------|--------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| <code>status</code>          | string<br><br>enum: IN_PROGRESS   CANCELED   SUCCEEDED | An optional filter that lets you search for jobs that have the specified status.                         |
| <code>targetSelection</code> | string<br><br>enum: CONTINUOUS   SNAPSHOT              | Specifies whether the job continues to run (CONTINUOUS), or is complete after all those things specified |

| Name           | Type                                                                     | Description                                                                                                                                                                                                                                                                                  |
|----------------|--------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                |                                                                          | as targets have completed the job (SNAPSHOT). If continuous, the job can 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. |
| maxResults     | integer<br><br>java class: java.lang.Integer<br><br>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<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9:_-]+       | A filter that limits the returned jobs to those for the specified group.                                                                                                                                                                                                                     |
| thingGroupId   | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9:_-]+       | A filter that limits the returned jobs to those for the specified group.                                                                                                                                                                                                                     |

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<br><br>member: JobSummary | A list of jobs. |

| Name            | Type                                                              | Description                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------|-------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 | java class: java.util.List                                        |                                                                                                                                                                                                                                                                                                                                                                                                       |
| JobSummary      | JobSummary                                                        |                                                                                                                                                                                                                                                                                                                                                                                                       |
| jobArn          | string                                                            | The job ARN.                                                                                                                                                                                                                                                                                                                                                                                          |
| jobId           | string<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+  | The unique identifier you assigned to this job when it was created.                                                                                                                                                                                                                                                                                                                                   |
| thingGroupId    | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9_-]+ | The ID of the thing group.                                                                                                                                                                                                                                                                                                                                                                            |
| targetSelection | string<br><br>enum: CONTINUOUS   SNAPSHOT                         | 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 can 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. |
| status          | string<br><br>enum: IN_PROGRESS   CANCELED   SUCCEEDED            | The job summary status.                                                                                                                                                                                                                                                                                                                                                                               |
| createdAt       | timestamp                                                         | The time, in seconds since the epoch, when the job was created.                                                                                                                                                                                                                                                                                                                                       |
| lastUpdatedAt   | timestamp                                                         | The time, in seconds since the epoch, when the job was last updated.                                                                                                                                                                                                                                                                                                                                  |
| completedAt     | timestamp                                                         | The time, in seconds since the epoch, when the job completed.                                                                                                                                                                                                                                                                                                                                         |
| nextToken       | string                                                            | The token for the next set of results, or <b>null</b> if there are no additional results.                                                                                                                                                                                                                                                                                                             |

## MQTT (12)

Not available.

## UpdateJob

### UpdateJob Command

Updates supported fields of the specified job. Updated values for `timeoutConfig` take effect for only newly in-progress executions. Currently in-progress executions continue to execute with the old timeout configuration.

HTTPS (13)

Request:

```
PATCH /jobs/jobId
{
    "description": "string",
    "presignedUrlConfig": {
        "expiresInSec": number,
        "roleArn": "string"
    },
    "jobExecutionsRolloutConfig": {
        "exponentialRate": {
            "baseRatePerMinute": number,
            "incrementFactor": number,
            "rateIncreaseCriteria": {
                "numberOfNotifiedThings": number,
                "numberOfSucceededThings": number
            },
            "maximumPerMinute": number
        },
        "abortConfig": {
            "criteriaList": [
                {
                    "action": "string",
                    "failureType": "string",
                    "minNumberOfExecutedThings": number,
                    "thresholdPercentage": number
                }
            ]
        },
        "timeoutConfig": {
            "inProgressTimeoutInMinutes": number
        }
    }
}
```

**jobId**

A job identifier that must be unique for your AWS account. We recommend using a UUID. Alphanumeric characters, "-", and "\_" can be used here.

**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 [Create Jobs \(p. 385\)](#).

`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 the AWS IoT Jobs service receives an MQTT request for the job document.

`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 are not more than the specified value.

`exponentialRate`

Allows you to create an exponential rate of rollout for a job.

`baseRatePerMinute`

The minimum number of things that are notified of a pending job, per minute at the start of job rollout. This parameter allows you to define the initial rate of rollout.

`incrementFactor`

The exponential factor to increase the rate of rollout for a job.

`rateIncreaseCriteria`

The criteria to initiate the increase in rate of rollout for a job. Set values for either the `numberOfNotifiedThings` or `numberOfSucceededThings`, but not both.

`numberOfNotifiedThings`

The threshold for number of notified things that initiate the increase in rate of rollout.

`numberOfSucceededThings`

The threshold for number of succeeded things that initiate the increase in rate of rollout.

`abortConfig`

Optional. Details of abort criteria to abort the job.

`criteriaList`

The list of abort criteria to define rules to abort the job.

`action`

The type of abort action to initiate a job abort.

`failureType`

The type of job execution failure to define a rule to initiate a job abort.

`minNumberOfExecutedThings`

Minimum number of executed things before evaluating an abort rule.

`thresholdPercentage`

The threshold as a percentage of the total number of executed things that initiate a job abort.

#### `timeoutConfig`

Optional. Specifies the amount of time each device has to finish its execution of the job. The timer is started when the job execution status is set to `IN_PROGRESS`. If the job execution status is not set to another terminal state before the time expires, it is set to `TIMED_OUT`.

#### `inProgressTimeoutInMinutes`

Specifies the amount of time, in minutes, this device has to finish execution of this job. A timer is started, or restarted, whenever this job's execution status is specified as `IN_PROGRESS` with this field populated. If the job execution status is not set to a terminal state before the timer expires, or before another job execution status update is sent with this field populated, the status is set to `TIMED_OUT`.

#### Response:

```
HTTP/1.1 200
```

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

CLI (13)

#### Synopsis:

```
aws iot update-job \
--job-id <value> \
[--description <value>] \
[--presigned-url-config <value>] \
[--job-executions-rollout-config <value>] \
[--abort-config <value>] \
[--timeout-config <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

#### cli-input-json format:

```
{
  "description": "string",
  "presignedUrlConfig": {
    "expiresInSec": number,
    "roleArn": "string"
  },
  "jobExecutionsRolloutConfig": {
    "exponentialRate": {
      "baseRatePerMinute": number,
      "incrementFactor": number,
      "rateIncreaseCriteria": {
        "numberOfNotifiedThings": number,
        "numberOfSucceededThings": number
      }
    },
    "maximumPerMinute": number
  },
  "abortConfig": {
    "criteriaList": [
      {
        "action": "string",
        "failureType": "string",
        "minNumberOfExecutedThings": number,
        "thresholdPercentage": number
      }
    ]
  },
}
```

```

    "timeoutConfig": {
        "inProgressTimeoutInMinutes": number
    }
}

```

**cli-input-json fields:**

| Name                       | Type                                                                      | Description                                                                                                                                                                                                              |
|----------------------------|---------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| jobId                      | string<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+          | A job identifier that must be unique for your AWS account. We recommend using a UUID. Alphanumeric characters, "-" and "_" are valid for use here.                                                                       |
| description                | string<br><br>length max:2028<br><br>pattern: [^\\p{C}]+                  | A short text description of the job.                                                                                                                                                                                     |
| presignedUrlConfig         | PresignedUrlConfig                                                        | Configuration information for presigned S3 URLs.                                                                                                                                                                         |
| roleArn                    | string<br><br>length max:2048 min:20                                      | The ARN of an IAM role that grants permission to download files from the S3 bucket where the job data or updates are stored. The role must also grant permission for AWS IoT to download the files.                      |
| expiresInSec               | long<br><br>java class: java.lang.Long<br><br>range- max:3600 min:60      | How long (in seconds) presigned URLs are valid. Valid values are 60 - 3600. The default value is 3600 seconds. Presigned URLs are generated when the AWS IoT Jobs service receives an MQTT request for the job document. |
| jobExecutionsRolloutConfig | JobExecutionsRolloutConfig                                                | Allows you to create a staged rollout of the job.                                                                                                                                                                        |
| maximumPerMinute           | integer<br><br>java class: java.lang.Integer<br><br>range- max:1000 min:1 | The maximum number of things that are notified of a pending job, per minute. This parameter allows you to create a staged rollout.                                                                                       |
| exponentialRate            | ExponentialRolloutRate                                                    | The rate of increase for a job rollout. This parameter allows you to define an exponential rate for a job rollout.                                                                                                       |
| baseRatePerMinute          | java class: java.lang.Integer                                             | The minimum number of things that are notified of a pending job, per minute at the start of job rollout. This                                                                                                            |

| Name                      | Type                                                               | Description                                                                                                                                                                                               |
|---------------------------|--------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           |                                                                    | parameter allows you to define the initial rate of rollout.                                                                                                                                               |
| incrementFactor           | java class: java.lang.Double                                       | The exponential factor to increase the rate of rollout for a job.                                                                                                                                         |
| rateIncreaseCriteria      | RateIncreaseCriteria                                               | Allows you to define a criteria to initiate the increase in rate of rollout for a job. Set a value for either <code>numberOfNotifiedThings</code> or <code>numberOfSucceededThings</code> , but not both. |
| numberOfNotifiedThings    | java class: java.lang.Double                                       | The threshold for number of notified things that initiate the increase in rate of rollout.                                                                                                                |
| numberOfSucceededThings   | java class: java.lang.Double                                       | The threshold for number of succeeded things that initiate the increase in rate of rollout.                                                                                                               |
| abortConfig               | AbortConfig                                                        | Allows you to create criteria to abort a job.                                                                                                                                                             |
| criteriaList              | AbortCriteria                                                      | The list of abort criteria to define rules to abort the job.                                                                                                                                              |
| action                    | java class: java.lang.String (CANCEL)                              | The type of abort action to initiate a job abort.                                                                                                                                                         |
| failureType               | java class: java.lang.String (FAILED   REJECTED   TIMED_OUT   ALL) | The type of job execution failure to define a rule to initiate a job abort.                                                                                                                               |
| minNumberOfExecutedThings | java class: java.lang.Integer                                      | Minimum number of executed things before evaluating an abort rule.                                                                                                                                        |
| thresholdPercentage       | java class: java.lang.Double                                       | The threshold as a percentage of the total number of executed things that initiate a job abort.<br><br>AWS IoT supports up to two digits after the decimal (for example, 10.9 and 10.99, but not 10.999). |

| Name                       | Type                                                               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------------|--------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| timeoutConfig              | TimeoutConfig                                                      | Specifies the amount of time each device has to finish its execution of the job. The timer is started when the job execution status is set to <code>IN_PROGRESS</code> . If the job execution status is not set to another terminal state before the time expires, it is set to <code>TIMED_OUT</code> .                                                                                                                                                  |
| inProgressTimeoutInMinutes | long                                                               | Specifies the amount of time, in minutes, this device has to finish execution of this job. A timer is started, or restarted, whenever this job's execution status is specified as <code>IN_PROGRESS</code> with this field populated. If the job execution status is not set to a terminal state before the timer expires, or before another job execution status update is sent with this field populated, the status is set to <code>TIMED_OUT</code> . |
| documentParameters         | map<br><br>key: ParameterKey<br><br>value: ParameterValue          | Parameters for the job document.                                                                                                                                                                                                                                                                                                                                                                                                                          |
| ParameterKey               | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9:_-]+ |                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| ParameterValue             | string<br><br>length max:1024 min:1<br><br>pattern: [^\p{C}]+      |                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

**Output:**

|              |
|--------------|
| HTTP/1.1 200 |
|--------------|

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

MQTT (13)

Not available.

# Jobs Device MQTT and HTTPS APIs

## Device MQTT and HTTPS Data Types

The following data types are used to communicate with the AWS IoT Jobs service 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|SUCCEEDED|CANCELED|TIMED_OUT|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, SUCCEEDED, CANCELED, TIMED\_OUT, REJECTED, or REMOVED.

###### statusDetails

A collection of name-value pairs that describe the status of the job execution.

###### queuedAt

The time, in seconds since the epoch, when the job execution was enqueued.

###### startedAt

The time, in seconds since the epoch, when the job execution was started.

`lastUpdatedAt`

The time, in seconds 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 job execution on a 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|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|  
    REMOVED",  
    "statusDetails": {  
        "string": "string"  
        ...  
    }  
    "versionNumber": "number"  
}
```

Description (8)

`status`

The status of the job execution. Can be one of: QUEUED, IN\_PROGRESS, FAILED, SUCCEEDED, CANCELED, TIMED\_OUT, 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 seconds since the epoch, when the job execution was enqueued.

**startedAt**

The time, in seconds since the epoch, when the job execution started.

**lastUpdatedAt**

The time, in seconds since the epoch, when the job execution was last updated.

**versionNumber**

The version of the job execution. Job execution versions are incremented each time the AWS IoT Jobs service receives an update from a device.

**executionNumber**

A number that identifies a job execution on a device.

## ErrorResponse

### ErrorResponse Data Type

Contains information about an error that occurred during an AWS IoT Jobs service 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 AWS IoT 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 `SUCCEEDED` 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 the AWS IoT Jobs service. In this case, the body of the error message also contains the `executionState` field.

#### InternalError

There was an internal error during the processing of 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 seconds since the epoch.

#### executionState

A [JobExecutionState \(p. 461\)](#) 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 state.

## MQTT (12)

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 is 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"
}
```

**inProgressJobs**

A list of [JobExecutionSummary \(p. 461\)](#) objects with status IN\_PROGRESS.

**queuedJobs**

A list of [JobExecutionSummary \(p. 461\)](#) objects with status QUEUED.

**clientToken**

A client token used to correlate requests and responses.

**timestamp**

The time, in seconds since the epoch, when the message was sent.

## HTTPS (12)

Request:

```
GET /things/thingName/jobs
```

**thingName**

The name of the thing associated with the device.

Response:

```
{
    "inProgressJobs" : [ JobExecutionSummary ... ],
    "queuedJobs" : [ JobExecutionSummary ... ]
}
```

#### inProgressJobs

A list of [JobExecutionSummary \(p. 461\)](#) objects.

#### queuedJobs

A list of [JobExecutionSummary \(p. 461\)](#) objects.

### CLI (12)

#### Synopsis:

```
aws iot-jobs-data 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<br><br>length max:128 min:1<br><br>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
        }
    ]
}
```

```

        "executionNumber": long
    }
]
}
}
```

**CLI output fields:**

| Name                | Type                                                                         | Description                                                                                                                                   |
|---------------------|------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| inProgressJobs      | list<br><br>member:<br>JobExecutionSummary<br><br>java class: java.util.List | A list of JobExecutionSummary objects with status IN_PROGRESS.                                                                                |
| JobExecutionSummary | JobExecutionSummary                                                          |                                                                                                                                               |
| jobId               | string<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+             | The unique identifier you assigned to this job when it was created.                                                                           |
| queuedAt            | long                                                                         | The time, in seconds since the epoch, when the job execution was enqueued.                                                                    |
| startedAt           | long<br><br>java class: java.lang.Long                                       | The time, in seconds since the epoch, when the job execution started.                                                                         |
| lastUpdatedAt       | long                                                                         | The time, in seconds 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 the AWS IoT Jobs service receives an update from a device. |
| executionNumber     | long<br><br>java class: java.lang.Long                                       | A number that identifies a job execution on a device.                                                                                         |
| queuedJobs          | list<br><br>member:<br>JobExecutionSummary<br><br>java class: java.util.List | A list of JobExecutionSummary objects with status QUEUED.                                                                                     |
| JobExecutionSummary | JobExecutionSummary                                                          |                                                                                                                                               |
| jobId               | string<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+             | The unique identifier you assigned to this job when it was created.                                                                           |

| Name            | Type                               | Description                                                                                                                                   |
|-----------------|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| queuedAt        | long                               | The time, in seconds since the epoch, when the job execution was enqueued.                                                                    |
| startedAt       | long<br>java class: java.lang.Long | The time, in seconds since the epoch, when the job execution started.                                                                         |
| lastUpdatedAt   | long                               | The time, in seconds 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 the AWS IoT Jobs service receives an update from a device. |
| executionNumber | long<br>java class: java.lang.Long | A number that identifies a job execution on a device.                                                                                         |

## StartNextPendingJobExecution

### StartNextPendingJobExecution Command

Gets and starts the next pending job execution for a thing (status IN\_PROGRESS or QUEUED).

- 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.
- You can optionally create a step timer by setting a value for the `stepTimeoutInMinutes` property. If you don't update the value of this property by running `UpdateJobExecution`, the job execution times out when the step timer expires.

### MQTT (13)

To invoke this API, publish a message on `$aws/things/thingName/jobs/start-next`.

Request payload:

```
{
    "statusDetails": {
        "string": "job-execution-state"
        ...
    },
    "stepTimeoutInMinutes": long,
    "clientToken": "string"
}
```

**statusDetails**

A collection of name-value pairs that describe the status of the job execution. If not specified, the `statusDetails` are unchanged.

**stepTimeOutInMinutes**

Specifies the amount of time this device has to finish execution of this job. If the job execution status is not set to a terminal state before this timer expires, or before the timer is reset (by calling `UpdateJobExecution`, setting the status to `IN_PROGRESS` and specifying a new timeout value in field `stepTimeoutInMinutes`) the job execution status is set to `TIMED_OUT`. Setting this timeout has no effect on that job execution timeout that might have been specified when the job was created (`CreateJob` using the `timeoutConfig` field).

**clientToken**

A client token used to correlate requests and responses. Enter an arbitrary value here and it is 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. 460) 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 (13)

Request:

```
PUT /things/thingName/jobs/$next
{
    "statusDetails": {
        "string": "string"
        ...
    },
    "stepTimeoutInMinutes": long
}
```

**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.

**stepTimeOutInMinutes**

Specifies the amount of time this device has to finish execution of this job. If the job execution status is not set to a terminal state before this timer expires, or before the timer is reset (by calling `UpdateJobExecution`, setting the status to `IN_PROGRESS` and specifying a new timeout value in field `stepTimeoutInMinutes`) the job execution status is set to `TIMED_OUT`. Setting this timeout has no effect on that job execution timeout that might have been specified when the job was created (`CreateJob` using the `timeoutConfig` field).

Response:

```
{
    "execution" : JobExecution
}
```

**execution**

A [JobExecution](#) (p. 460) 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,
}
```

## CLI (13)

**Synopsis:**

```
aws iot-jobs-data start-next-pending-job-execution \
--thing-name <value> \
[--step-timeout-in-minutes <value>] \
[--status-details <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

**cli-input-json** format:

```
{
  "thingName": "string",
  "statusDetails": {
    "string": "string"
  },
  "stepTimeoutInMinutes": long
}
```

**cli-input-json fields:**

| Name                 | Type                                                       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| thingName            | string<br>length max:128 min:1<br>pattern: [a-zA-Z0-9:_-]+ | The name of the thing associated with the device.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| statusDetails        | map<br>key: DetailsKey<br>value: DetailsValue              | A collection of name-value pairs that describe the status of the job execution. If not specified, the statusDetails are unchanged.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| stepTimeOutInMinutes | long                                                       | Specifies the amount of time this device has to finish execution of this job. If the job execution status is not set to a terminal state before this timer expires, or before the timer is reset (by calling <a href="#">UpdateJobExecution</a> , setting the status to <a href="#">IN_PROGRESS</a> and specifying a new timeout value in field <code>stepTimeoutInMinutes</code> ) the job execution status is set to <a href="#">TIMED_OUT</a> . Setting this timeout has no effect on that job execution timeout that might have been specified when the job was created ( <a href="#">CreateJob</a> using the <code>timeoutConfig</code> field). |
| DetailsKey           | string<br>length max:128 min:1<br>pattern: [a-zA-Z0-9:_-]+ |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

| Name         | Type                                                           | Description |
|--------------|----------------------------------------------------------------|-------------|
| DetailsValue | string<br><br>length max:1024 min:1<br><br>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<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+                                          | The unique identifier you assigned to this job when it was created.                                                                |
| thingName     | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9:_-]+                                        | The name of the thing that is executing the job.                                                                                   |
| status        | string<br><br>enum: QUEUED   IN_PROGRESS   SUCCEEDED   FAILED   TIMED_OUT   REJECTED   REMOVED   CANCELED | The status of the job execution. Can be one of: QUEUED, IN_PROGRESS, FAILED, SUCCEEDED, CANCELED, TIMED_OUT, REJECTED, or REMOVED. |
| statusDetails | map<br><br>key: DetailsKey<br><br>value: DetailsValue                                                     | A collection of name-value pairs that describe the status of the job execution.                                                    |
| DetailsKey    | string                                                                                                    |                                                                                                                                    |

| Name            | Type                                                            | Description                                                                                                                             |
|-----------------|-----------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
|                 | length max:128 min:1<br>pattern: [a-zA-Z0-9:_-]+                |                                                                                                                                         |
| DetailsValue    | string<br><br>length max:1024 min:1<br><br>pattern: [^\\p{C}]*+ |                                                                                                                                         |
| queuedAt        | long                                                            | The time, in seconds since the epoch, when the job execution was enqueued.                                                              |
| startedAt       | long<br><br>java class: java.lang.Long                          | The time, in seconds since the epoch, when the job execution was started.                                                               |
| lastUpdatedAt   | long                                                            | The time, in seconds 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<br><br>java class: java.lang.Long                          | A number that identifies a job execution on a device. It can be used later in commands that return or update job execution information. |
| jobDocument     | string<br><br>length max:32768                                  | The content of the job document.                                                                                                        |

## DescribeJobExecution

### DescribeJobExecution Command

Gets detailed information about a job execution.

You can set the `jobId` to `$next` to return the next pending job execution for a thing (status `IN_PROGRESS` or `QUEUED`).

#### MQTT (14)

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 job execution for a thing (status IN\_PROGRESS or QUEUED). 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 job execution on a device. If not specified, the latest job execution is returned.

`includeJobDocument`

Optional. Unless set to `false`, 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 is 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 \(p. 460\)](#) object.

`timestamp`

The time, in seconds since the epoch, when the message was sent.

`clientToken`

A client token used to correlate requests and responses.

## HTTPS (14)

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 job execution for a thing (status IN\_PROGRESS or QUEUED). 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. Unless set to `false`, the response contains the job document. The default is `true`.

`executionNumber`

Optional. A number that identifies a job execution on a device. If not specified, the latest job execution is returned.

**Response:**

```
{
    "execution" : JobExecution,
}
```

`execution`

A [JobExecution \(p. 460\)](#) object.

## CLI (14)

The job's execution status must be QUEUED or IN\_PROGRESS.

### Synopsis:

```
aws iot-jobs-data 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                                                                                               |
|--------------------|-----------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| <code>jobId</code> | string<br><br>pattern: [a-zA-Z0-9_-]+ +\$next | The unique identifier assigned to this job when it was created, or <code>\$next</code> to return the next |

| Name               | Type                                                      | Description                                                                                                                                                                                                         |
|--------------------|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                    |                                                           | pending job execution for a thing (status IN_PROGRESS or QUEUED). 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. |
| thingName          | string<br>length max:128 min:1<br>pattern: [a-zA-Z0-9:_]+ | The thing name associated with the device the job execution is running on.                                                                                                                                          |
| includeJobDocument | boolean<br>java class: java.lang.Boolean                  | Optional. Unless set to false, the response contains the job document. The default is true.                                                                                                                         |
| executionNumber    | long<br>java class: java.lang.Long                        | Optional. A number that identifies a job execution on a 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<br>length max:64 min:1<br>pattern: [a-zA-Z0-9:_]+ | The unique identifier you assigned to this job when it was created. |
| thingName | string                                                   | The name of the thing that is executing the job.                    |

| Name            | Type                                                                                                      | Description                                                                                                                             |
|-----------------|-----------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
|                 | length max:128 min:1<br><br>pattern: [a-zA-Z0-9:_-]+                                                      |                                                                                                                                         |
| status          | string<br><br>enum: QUEUED   IN_PROGRESS   SUCCEEDED   FAILED   TIMED_OUT   REJECTED   REMOVED   CANCELED | The status of the job execution. Can be one of: QUEUED, IN_PROGRESS, FAILED, SUCCEEDED, CANCELED, TIMED_OUT, REJECTED, or REMOVED.      |
| statusDetails   | map<br><br>key: DetailsKey<br><br>value: DetailsValue                                                     | A collection of name-value pairs that describe the status of the job execution.                                                         |
| DetailsKey      | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9:_-]+                                        |                                                                                                                                         |
| DetailsValue    | string<br><br>length max:1024 min:1<br><br>pattern: [^\p{C}]*+                                            |                                                                                                                                         |
| queuedAt        | long                                                                                                      | The time, in seconds since the epoch, when the job execution was enqueued.                                                              |
| startedAt       | long<br><br>java class: java.lang.Long                                                                    | The time, in seconds since the epoch, when the job execution was started.                                                               |
| lastUpdatedAt   | long                                                                                                      | The time, in seconds 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<br><br>java class: java.lang.Long                                                                    | A number that identifies a job execution on a device. It can be used later in commands that return or update job execution information. |
| jobDocument     | string<br><br>length max:32768                                                                            | The content of the job document.                                                                                                        |

## UpdateJobExecution

### UpdateJobExecution Command

Updates the status of a job execution. You can optionally create a step timer by setting a value for the `stepTimeoutInMinutes` property. If you don't update the value of this property by running `UpdateJobExecution` again, the job execution times out when the step timer expires.

MQTT (15)

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,  
    "stepTimeoutInMinutes": long,  
    "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 the AWS IoT Jobs service does not match, the update is rejected with a `VersionMismatch` error, and an [ErrorResponse \(p. 462\)](#) that contains the current job execution status data is returned. (This makes it unnecessary to perform a separate `DescribeJobExecution` request to obtain the job execution status data.)

**executionNumber**

Optional. A number that identifies a job execution on a 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`.

**stepTimeoutInMinutes**

Specifies the amount of time this device has to finish execution of this job. If the job execution status is not set to a terminal state before this timer expires, or before the timer is reset (by

again calling `UpdateJobExecution`, setting the status to `IN_PROGRESS` and specifying a new timeout value in this field) the job execution status is set to `TIMED_OUT`. Setting or resetting this timeout has no effect on the job execution timeout that might have been specified when the job was created (by using `CreateJob` with the `timeoutConfig`).

`clientToken`

A client token used to correlate requests and responses. Enter an arbitrary value here and it is 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 \(p. 461\)](#) object.

`jobDocument`

A [job document \(p. 381\)](#) object.

`timestamp`

The time, in seconds since the epoch, when the message was sent.

`clientToken`

A client token used to correlate requests and responses.

## HTTPS (15)

Request:

```
POST /things/thingName/jobs/jobId  
{  
    "status": "job-execution-state",  
    "statusDetails": {  
        "string": "string"  
        ...  
    },  
    "expectedVersion": "number",  
    "includeJobExecutionState": boolean,  
    "includeJobDocument": boolean,  
    "stepTimeoutInMinutes": long,  
    "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 the AWS IoT Jobs service does not match, the update is rejected with a `VersionMismatch` error, and an [ErrorResponse \(p. 462\)](#) that contains the current job execution status data is returned. (This makes it unnecessary to perform a separate `DescribeJobExecution` request 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`.

`stepTimeoutInMinutes`

Specifies the amount of time this device has to finish execution of this job. If the job execution status is not set to a terminal state before this timer expires, or before the timer is reset (by again calling `UpdateJobExecution`, setting the status to IN\_PROGRESS and specifying a new timeout value in this field) the job execution status is set to TIMED\_OUT. Setting or resetting this timeout has no effect on the job execution timeout that might have been specified when the job was created (by using `CreateJob` with the `timeoutConfig`).

`executionNumber`

Optional. A number that identifies a job execution on a device.

Response:

```
{  
    "executionState": JobExecutionState,  
    "jobDocument": "string"  
}
```

`executionState`

A [JobExecutionState \(p. 461\)](#) object.

`jobDocument`

The contents of the [job document \(p. 381\)](#).

## CLI (15)

### Synopsis:

```
aws iot-jobs-data 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>] \
    [--step-timeout-in-minutes <value>] \
    [--generate-cli-skeleton]
```

**cli-input-json** format:

```
{
    "jobId": "string",
    "thingName": "string",
    "status": "string",
    "statusDetails": {
        "string": "string"
    },
    "stepTimeoutInMinutes": number,
    "expectedVersion": long,
    "includeJobExecutionState": boolean,
    "includeJobDocument": boolean,
    "executionNumber": long
}
```

**cli-input-json** fields:

| Name          | Type                                                                                                      | Description                                                                                                                 |
|---------------|-----------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| jobId         | string<br><br>length max:64 min:1<br><br>pattern: [a-zA-Z0-9_-]+                                          | The unique identifier assigned to this job when it was created.                                                             |
| thingName     | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9_-]+                                         | The name of the thing associated with the device.                                                                           |
| status        | string<br><br>enum: QUEUED   IN_PROGRESS   SUCCEEDED   FAILED   TIMED_OUT   REJECTED   REMOVED   CANCELED | The new status for the job execution (IN_PROGRESS, FAILED, SUCCEEDED, or REJECTED). This must be specified on every update. |
| statusDetails | map<br><br>key: DetailsKey<br><br>value: DetailsValue                                                     | Optional. A collection of name-value pairs that describe the status of the job execution. If                                |

| Name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Type                                                              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                   | not specified, the statusDetails are unchanged.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| DetailsKey                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9:_]+ |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| DetailsValue                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | string<br><br>length max:1024 min:1<br><br>pattern: [^\p{C}]*+    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| stepTimeoutInMinutes<br><br>long<br><br>Specifies the amount of time this device has to finish execution of this job. If the job execution status is not set to a terminal state before this timer expires, or before the timer is reset (by again calling <code>UpdateJobExecution</code> , setting the status to <code>IN_PROGRESS</code> and specifying a new timeout value in this field) the job execution status is set to <code>TIMED_OUT</code> . Setting or resetting this timeout has no effect on the job execution timeout that might have been specified when the job was created (by using <code>CreateJob</code> with the <code>timeoutConfig</code> ). |                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| expectedVersion                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | long<br><br>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 the AWS IoT Jobs service does not match, the update is rejected with a <code>VersionMismatch</code> error, and an <code>ErrorResponse</code> that contains the current job execution status data is returned. (This makes it unnecessary to perform a separate <code>DescribeJobExecution</code> request to obtain the job execution status data.) |

| Name                     | Type                                     | Description                                                                                                      |
|--------------------------|------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| includeJobExecutionState | boolean<br>java class: java.lang.Boolean | Optional. When included and set to true, the response contains the JobExecutionState data. The default is false. |
| includeJobDocument       | boolean<br>java class: java.lang.Boolean | Optional. When set to true, the response contains the job document. The default is false.                        |
| executionNumber          | long<br>java class: java.lang.Long       | Optional. A number that identifies a job execution on a device.                                                  |

Output:

```
{
  "executionState": {
    "status": "string",
    "statusDetails": {
      "string": "string"
    },
    "versionNumber": long
  },
  "jobDocument": "string"
}
```

#### CLI output fields:

| Name           | Type                                                                                                      | Description                                                                                                                        |
|----------------|-----------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| executionState | JobExecutionState                                                                                         | A JobExecutionState object.                                                                                                        |
| status         | string<br><br>enum: QUEUED   IN_PROGRESS   SUCCEEDED   FAILED   TIMED_OUT   REJECTED   REMOVED   CANCELED | The status of the job execution. Can be one of: QUEUED, IN_PROGRESS, FAILED, SUCCEEDED, CANCELED, TIMED_OUT, REJECTED, or REMOVED. |
| statusDetails  | map<br><br>key: DetailsKey<br><br>value: DetailsValue                                                     | A collection of name-value pairs that describe the status of the job execution.                                                    |
| DetailsKey     | string<br><br>length max:128 min:1<br><br>pattern: [a-zA-Z0-9:_-]+                                        |                                                                                                                                    |
| DetailsValue   | string<br><br>length max:1024 min:1<br><br>pattern: [^\p{C}]*+                                            |                                                                                                                                    |

| Name          | Type                       | Description                                                                                                      |
|---------------|----------------------------|------------------------------------------------------------------------------------------------------------------|
| versionNumber | long                       | The version of the job execution. Job execution versions are incremented each time they are updated by a device. |
| jobDocument   | string<br>length max:32768 | The contents of the job documents.                                                                               |

## JobExecutionsChanged

### JobExecutionsChanged Message

Sent whenever a job execution is added to or removed from the list of pending job executions for a thing.

#### MQTT (16)

Topic: `$aws/things/thingName/jobs/notify`

Message payload:

```
{
  "jobs" : [
    "JobExecutionState": [ JobExecutionSummary \(p. 410\) ... ]
  ],
  "timestamp": timestamp,
}
```

#### HTTPS (16)

Not available.

#### CLI (16)

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 \(p. 472\)](#) 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 (17)

Topic: `$aws/things/thingName/jobs/notify-next`

Message payload:

```
{
  "execution" : JobExecution \(p. 409\),
```

```
    "timestamp": timestamp,  
}
```

#### HTTPS (17)

Not available.

#### CLI (17)

Not available.

## Job Rollout and Abort Configuration

AWS IoT jobs can be deployed using variable rollout rates as various criteria and thresholds are met. Job rollouts can also be aborted if the number of failed jobs matches a set of criteria. These rollout configurations give you more granular control over a job's blast radius. Job rollout rate criteria are set at job creation through the [JobExecutionsRolloutConfig](#) object. Job abort criteria are set at job creation through the [AbortConfig](#) object.

## Using Job Rollout Rates

You set a job rollout rate by configuring the [ExponentialRolloutRate](#) property of the [JobExecutionsRolloutConfig](#) object when you run the [CreateJob](#) API. The following example sets a variable rollout rate by using the `exponentialRate` parameter.

```
{  
...  
    "jobExecutionsRolloutConfig": {  
        "exponentialRate": {  
            "baseRatePerMinute": 50,  
            "incrementFactor": 2,  
            "rateIncreaseCriteria": {  
                "numberOfNotifiedThings": 1000, // Set one or the other  
                "numberOfSucceededThings": 1000 // of these two values.  
            },  
            "maximumPerMinute": 1000  
        }  
    }  
}
```

The `baseRatePerMinute` parameter specifies the rate at which the jobs are executed until the `numberOfNotifiedThings` or `numberOfSucceededThings` threshold has been met.

The `incrementFactor` parameter specifies the exponential factor by which the rollout rate increases after the `numberOfNotifiedThings` or `numberOfSucceededThings` threshold has been met.

The `rateIncreaseCriteria` parameter is an object that specifies either the `numberOfNotifiedThings` or `numberOfSucceededThings` threshold.

The `maximumPerMinute` parameter specifies the upper limit of the rate at which job executions can occur. Valid values range from 1 to 1000. This parameter is required when you pass an [ExponentialRate](#) object. In a variable rate rollout, this value establishes the upper limit of a job rollout rate.

A job rollout with the configuration above would start at a rate of 50 job executions per minute. It would continue at that rate until either 1000 things have received job execution notifications (if a value for

`numberOfNotifiedThings` has been specified) or 1000 successful job executions have occurred (if a value for `numberOfSucceededThings` has been specified).

The following table illustrates how the rollout would proceed over the first four increments.

| Rollout rate per minute                             | 50   | 100  | 200  | 400  |
|-----------------------------------------------------|------|------|------|------|
| Number of notified devices or successful executions | 1000 | 2000 | 3000 | 4000 |

The following configuration sets a static rollout rate.

```
{
...
"jobExecutionsRolloutConfig": {
    "maximumPerMinute": 1000
}
}
```

The `maximumPerMinute` parameter specifies the upper limit of the rate at which job executions can occur. Valid values range from 1 to 1000. This parameter is optional. If you don't specify a value, the default value of 1000 is used.

## Using Job Rollout Abort Configurations

You set up a job abort condition by configuring the optional `AbortConfig` object when you run the `CreateJob` API. This section describes the effect that the following sample configuration would have on a job rollout that was experiencing multiple failed executions.

```
"abortConfig": {
    "criteriaList": [
        {
            "action": "CANCEL",
            "failureType": "FAILED",
            "minNumberOfExecutedThings": 100,
            "thresholdPercentage": 20
        },
        {
            "action": "CANCEL",
            "failureType": "TIMED_OUT",
            "minNumberOfExecutedThings": 200,
            "thresholdPercentage": 50
        }
    ]
}
```

The `action` parameter specifies the action to take when the abort criteria have been met. This parameter is required, and `CANCEL` is the only valid value.

The `failureType` parameter specifies which failure types should trigger a job abort. Valid values are `FAILED`, `REJECTED`, `TIMED_OUT`, and `ALL`.

The `minNumberOfExecutedThings` parameter specifies the number of completed job executions that must occur before the service checks to see if the job abort criteria have been met. In this example, AWS IoT doesn't check to see if a job abort should occur until at least 100 devices have completed job executions.

The `thresholdPercentage` parameter specifies the total number of executed things that initiate a job abort. In this example, AWS IoT initiates a job abort and cancels the job rollout if at least 20% of all completed executions have failed in any way after 100 executions have completed.

**Note**

Deletion of job executions affects the computation value of the total completed execution. When a job aborts, the service creates an automated comment and `reasonCode` to differentiate a user-driven cancellation from a job abort cancellation.

## Job Limits

For job limit information, see [AWS IoT Endpoints and Quotas](#) in the AWS General Reference.

# AWS IoT Secure Tunneling

When devices are deployed behind restricted firewalls at remote sites, you need a way to gain access to those devices for troubleshooting, configuration updates, and other operational tasks. Secure tunneling helps customers establish bidirectional communication to remote devices over a secure connection that is managed by AWS IoT. Secure tunneling does not require updates to your existing inbound firewall rule, so you can keep the same security level provided by firewall rules at a remote site.

For example, a sensor device located at a factory that is a couple hundred miles away is having trouble measuring the factory temperature. You can use secure tunneling to open and quickly start a session to that sensor device. After you have identified the problem (for example, a bad configuration file), you can reset the file and restart the sensor device through the same session. Compared to a more traditional troubleshooting (for example, sending a technician to the factory to investigate the sensor device), secure tunneling decreases incident response and recovery time and operational costs.

## Secure Tunneling Concepts

### Client access token (CAT)

A pair of tokens generated by secure tunneling when a new tunnel is created. The CAT is used by the source and destination devices to connect to the Secure Tunneling service.

### Destination application

The application that runs on the destination device. For example, the destination application can be an SSH daemon for establishing an SSH session using secure tunneling.

### Destination device

The remote device you want to access.

### Device agent

An IoT application that connects to the AWS IoT device gateway and listens for new tunnel notifications over MQTT.

### Local proxy

A software proxy that runs on the source and destination devices and relays a data stream between the Secure Tunneling service and the device application. The local proxy can be run in source mode or destination mode. For more information, see [Local Proxy \(p. 493\)](#).

### Source device

The device an operator uses to initiate a session to the destination device, usually a laptop or desktop computer.

### Tunnel

A logical pathway through AWS IoT that enables bidirectional communication between a source device and destination device.

## Secure Tunneling Tutorial

In this tutorial, you open a tunnel and use it to start an SSH session to a remote device. The remote device is behind firewalls that block all inbound traffic, making direct SSH into the device impossible.

Before you begin, make sure that you understand how to [register a device in the AWS IoT registry](#) and [connect a device to the AWS IoT device gateway](#).

## Prerequisites

- The firewalls the remote device is behind must allow outbound traffic on port 443.
- You have created an IoT thing named `RemoteDeviceA` in the AWS IoT registry.
- You have an IoT device agent running on the remote device that connects to the AWS IoT device gateway and is configured with an MQTT topic subscription. This tutorial includes a snippet that shows you how to implement an agent. For more information, see [IoT Agent Snippet \(p. 494\)](#).
- You must have an SSH daemon running on the remote device.
- You have downloaded the local proxy source code from [GitHub](#) and built it for the platform of your choice. We'll refer to the built local proxy executable file as `localproxy` in this tutorial.

## Open a Tunnel

1. Open the [AWS IoT console](#). In the navigation pane, choose **Manage**, and then choose **Tunnels**.
2. Choose **Open a tunnel**.
3. On the **Open a new tunnel** page, enter an optional description for the tunnel.
4. Under **Choose a thing to open a tunnel for**, choose `RemoteDeviceA`.
5. Under **Service**, enter `SSH`, and then choose **Open New**. For more information, see [IoT Agent Snippet \(p. 494\)](#).
6. In the **New tunnel opened** page, download your source and destination access tokens and save them to your computer. This is the only time you can retrieve the tokens.
7. Choose **Done**.

If you configure the destination when calling `OpenTunnel`, the Secure Tunneling service delivers the destination client access token to the remote device over MQTT and the reserved MQTT topic (`$aws/things/RemoteDeviceA/tunnels/notify`). For more information, [MQTT Reserved topics](#). Upon receipt of the MQTT message, the IoT agent on the remote device starts the local proxy in destination mode. You can omit the destination configuration if you want to deliver the destination client access token to the remote device through another method. For more information, see [Configuring a Remote Device \(p. 495\)](#).

## Start the Local Proxy

Open a terminal on your laptop, copy the source client access token, and use it to start the local proxy in source mode. In the following command, the local proxy is configured to listen for new connections on port 5555.

```
./localproxy -r us-east-1 -s 5555 -t <source-client-access-token>
```

### Note

The AWS Region in this command must be the same AWS Region where the tunnel was created.

-r

Specifies the AWS Region where your tunnel is created.

-s

Specifies the port to which the proxy should connect.

-t

Specifies the client token text.

**Note**

If you receive the following error, set up the CA path. For information, see [GitHub](#).

Could not perform SSL handshake with proxy server: certificate verify failed

## Start an SSH Session

Open another terminal and use the following command to start a new SSH session by connecting to the local proxy on port 5555.

```
ssh <username>@localhost -p 5555
```

You might be prompted for a password for the SSH session. When you are done with the SSH session, type `exit` to close the session.

## Close the Tunnel

1. Open the [AWS IoT console](#).
2. Choose the tunnel and from **Actions**, choose **Close**. Closing the tunnel causes both local proxy instances to close.

## Secure Tunnel Lifecycle

Tunnels can have one of the following statuses:

- OPEN
- CLOSED

Connections can have one of the following statuses:

- CONNECTED
- DISCONNECTED

When you open a tunnel, it has a status of OPEN. The tunnel's source and destination connection status is set to DISCONNECTED. When a device (source or destination) connects to the tunnel, the corresponding connection status changes to CONNECTED, while the tunnel status remains OPEN. When a device disconnects from the tunnel, the corresponding connection status changes back to DISCONNECTED. A device can connect to and disconnect from a tunnel repeatedly as long as the tunnel remains OPEN.

A tunnel's status becomes CLOSED when you call `CloseTunnel` or the tunnel remains OPEN for longer than the `MaxLifetimeTimeout` value. You can configure `MaxLifetimeTimeout` when calling `OpenTunnel`. `MaxLifetimeTimeout` defaults to 12 hours if you do not specify a value.

After a tunnel is CLOSED, it might be visible in the AWS IoT console for at least three hours before it is deleted. While the tunnel is visible, you can call `DescribeTunnel` and `ListTunnels` to view tunnel metadata. A tunnel cannot be reopened when it is CLOSED. The Secure Tunneling service rejects attempts to connect to a CLOSED tunnel.

# Controlling Access to Tunnels

The Secure Tunneling service provides the following service-specific actions, resources, and condition context keys for use in IAM permission policies.

## Tunnel Access Prerequisites

- Learn how to secure AWS resources by using [IAM policies](#).
- Learn how to create and evaluate [IAM conditions](#).
- Learn how to secure AWS resources using [resource tags](#).

### iot:OpenTunnel

The `iot:OpenTunnel` policy action grants a principal permission to call `OpenTunnel`. You must specify the wildcard tunnel ARN `arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/*` in the `Resource` element of the IAM policy statement. You can specify a thing ARN (`arn:aws:iot:<aws-region>:<aws-account-id>:thing/ <thing-name>`) in the `Resource` element of the IAM policy statement to manage `OpenTunnel` permission for specific IoT things.

For example, the following policy statement allows you to open a tunnel to the IoT thing named `TestDevice`.

```
{  
    "Effect": "Allow",  
    "Action": "iot:OpenTunnel",  
    "Resource": [  
        "arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/*",  
        "arn:aws:iot:<aws-region>:<aws-account-id>:thing/TestDevice"  
    ]  
}
```

The `iot:OpenTunnel` policy action supports the following condition keys:

- `iot:ThingGroupArn`
- `iot:TunnelDestinationService`
- `aws:RequestTag/<tag-key>`
- `aws:TagKeys`

The following policy statement allows you to open a tunnel to the thing if the thing belongs to a thing group with a name that starts with `TestGroup` and the configured destination service on the tunnel is `SSH`.

```
{  
    "Effect": "Allow",  
    "Action": "iot:OpenTunnel",  
    "Resource": [  
        "arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/*"  
    ],  
    "Condition": {  
        "ForAnyValue:StringLike": {  
            "iot:ThingGroupArn": [  
                "arn:aws:iot:<aws-region>:<aws-account-id>:thinggroup/TestGroup*"  
            ]  
        },  
        "ForAllValues:StringEquals": {  
            "aws:RequestTag/destinationService": "SSH"  
        }  
    }  
}
```

```

        "iot:TunnelDestinationService": [
            "SSH"
        ]
    }
}
}
```

You can also use resource tags to control permission to open tunnels. For example, the following policy statement allows a tunnel to be opened if the tag key `Owner` is present with a value of `Admin` and no other tags are specified.

```
{
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/Owner": "Admin"
        },
        "ForAllValues:StringEquals": {
            "aws:TagKeys": "Owner"
        }
    }
}
```

## iot:DescribeTunnel

The `iot:DescribeTunnel` policy action grants a principal permission to call [DescribeTunnel](#). You can specify a fully qualified tunnel ARN (for example, `arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/<tunnel-id>`) or use the wildcard tunnel ARN (`arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/*`) in the `Resource` element of the IAM policy statement.

The `iot:DescribeTunnel` policy action supports the following condition key:

- `aws:ResourceTag/<tag-key>`

The following policy statement allows you to call `DescribeTunnel` if the requested tunnel is tagged with the key `Owner` with a value of `Admin`.

```
{
    "Effect": "Allow",
    "Action": "iot:DescribeTunnel",
    "Resource": [
        "arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/Owner": "Admin"
        }
    }
}
```

## iot>ListTunnels

The `iot>ListTunnels` policy action grants a principal permission to call [ListTunnels](#). You must specify the wildcard tunnel ARN (`arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/`

\*) in the Resource element of the IAM policy statement. To manage ListTunnels permission on selected IoT things, you can also specify a thing ARN (arn:aws:iot:<aws-region>:<aws-account-id>:thing/<thing-name>) in the Resource element of the IAM policy statement.

The following policy statement allows you to list tunnels for the thing named TestDevice.

```
{
    "Effect": "Allow",
    "Action": "iot>ListTunnels",
    "Resource": [
        "arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/*",
        "arn:aws:iot:<aws-region>:<aws-account-id>:thing/TestDevice"
    ]
}
```

## iot>ListTagsForResource

The iot>ListTagsForResource policy action grants a principal permission to call ListTagsForResource. You can specify a fully qualified tunnel ARN (arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/<tunnel-id>) or use the wildcard tunnel ARN (arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/\*) in the Resource element of the IAM policy statement.

## iot:CloseTunnel

The iot:CloseTunnel policy action grants a principal permission to call CloseTunnel. You can specify a fully qualified tunnel ARN (arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/<tunnel-id>) or use the wildcard tunnel ARN (arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/\*) in the Resource element of the IAM policy statement.

The iot:CloseTunnel policy action supports the following condition keys:

- iot>Delete
- aws:ResourceTag/<tag-key>

The following policy statement allows you to call CloseTunnel if the request's Delete parameter is false and the requested tunnel is tagged with the key Owner with a value of QATeam.

```
{
    "Effect": "Allow",
    "Action": "iot:CloseTunnel",
    "Resource": [
        "arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/*"
    ],
    "Condition": {
        "Bool": {
            "iot>Delete": "false"
        },
        "StringEquals": {
            "aws:ResourceTag/Owner": "QATeam"
        }
    }
}
```

## iot:TagResource

The iot:TagResource policy action grants a principal permission to call TagResource. You can specify a fully qualified tunnel ARN (arn:aws:iot:<aws-region>:<aws-account-

`id>:tunnel/<tunnel-id>`) or use the wildcard tunnel ARN (`arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/*`) in the Resource element of the IAM policy statement.

## iot:UntagResource

The `iot:UntagResource` policy action grants a principal permission to call `UntagResource`. You can specify a fully qualified tunnel ARN (`arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/<tunnel-id>`) or use the wildcard tunnel ARN (`arn:aws:iot:<aws-region>:<aws-account-id>:tunnel/*`) in the Resource element of the IAM policy statement.

For more information about AWS IoT security see [Identity and Access Management for AWS IoT \(p. 174\)](#).

## Local Proxy

The local proxy is a process that acts as the recipient or sender of incoming TCP connections. It transmits data sent by the device application through the Secure Tunneling service over a WebSocket secure connection. You can download the local proxy source from [GitHub](#). The local proxy can run in two modes: source or destination. In source mode, the local proxy runs on the same device or network as the client application that initiates the TCP connection. In destination mode, the local proxy runs on the remote device, along with the destination application. Currently, a single tunnel can support only one TCP connection at a time.

The local proxy first establishes a connection to the Secure Tunneling service. When you start the local proxy, use the `-r` argument to specify the AWS Region in which the tunnel is opened. Use the `-t` argument to pass either the source or destination client access token returned from the `OpenTunnel`. Two local proxies using the same client access token value cannot be connected at the same time.

After the WebSocket connection is established, the local proxy performs either source mode or destination mode behaviors, depending on its configuration.

By default, the local proxy attempts to reconnect to the Secure Tunneling service if any I/O errors occur or if the WebSocket connection is closed unexpectedly. This causes the TCP connection to close. If any TCP socket errors occur, the local proxy sends a message through the tunnel to notify the other side to close its TCP connection. By default, the local proxy always uses SSL communication.

After you use the tunnel, it is safe to terminate the local proxy process. We recommend that you explicitly close the tunnel by calling `CloseTunnel`. Active tunnel clients might not be closed immediately after calling `CloseTunnel`.

## Local Proxy Security Best Practices

When running the local proxy, follow these security best practices:

- Avoid the use of the `-t` local proxy argument to pass in an access token. We recommend that you use the `AWSIOT_TUNNEL_ACCESS_TOKEN` environment variable to set the access token for the local proxy.
- Run the local proxy executable with least privileges in the operating system or environment.
  - Avoid running the local proxy as an administrator on Windows.
  - Avoid running the local proxy as root on Linux and macOS.
- Consider running the local proxy on separate hosts, containers, sandboxes, chroot jail or a virtualized environment.
- Build the local proxy with relevant security flags, depending on your toolchain.
- On devices with multiple network interfaces, use the `-b` argument to bind the TCP socket to the network interface used to communicate with the destination application.

# IoT Agent Snippet

The IoT agent is used to receive the MQTT message that includes the client access token and start a local proxy on the remote device. You must install and run the IoT agent on the remote device if you want the Secure Tunneling service to deliver the client access token. The IoT agent must subscribe to the following reserved IoT MQTT topic:

```
$aws/things/<thing-name>/tunnels/notify
```

Where thing-name is the name of IoT thing associated with the remote device.

The following is an example MQTT message payload:

```
{  
    "clientAccessToken": "<destination-client-access-token>",  
    "clientMode": "destination",  
    "region": "<aws-region>",  
    "services": ["<destination-service>"]  
}
```

After it receives an MQTT message, the IoT agent must start a local proxy on the remote device with the appropriate parameters.

The following Java code demonstrates how to use the [AWS IoT Device SDK](#) and [ProcessBuilder](#) from the Java library to build a simple IoT agent to work with the Secure Tunneling service.

```
// Find the IoT device endpoint for your AWS account  
final String endpoint = iotClient.describeEndpoint(new  
    DescribeEndpointRequest().withEndpointType("iot:Data-ATS")).getEndpointAddress();  
  
// Instantiate the IoT Agent with your AWS credentials  
final String thingName = "RemoteDeviceA";  
final String tunnelNotificationTopic = String.format("$aws/things/%s/tunnels/notify",  
    thingName);  
final AWSIotMqttClient mqttClient = new AWSIotMqttClient(endpoint, thingName,  
    "your_aws_access_key", "your_aws_secret_key");  
  
try {  
    mqttClient.connect();  
    final TunnelNotificationListener listener = new  
        TunnelNotificationListener(tunnelNotificationTopic);  
    mqttClient.subscribe(listener, true);  
}  
finally {  
    mqttClient.disconnect();  
}  
  
private static class TunnelNotificationListener extends AWSIotTopic {  
    public TunnelNotificationListener(String topic) {  
        super(topic);  
    }  
  
    @Override  
    public void onMessage(AWSIotMessage message) {  
        try {  
            // Deserialize the MQTT message  
            final JSONObject json = new JSONObject(message.getStringPayload());  
  
            final String accessToken = json.getString("clientAccessToken");  
            final String region = json.getString("region");  
        }  
    }  
}
```

```
final String clientMode = json.getString("clientMode");
if (!clientMode.equals("destination")) {
    throw new RuntimeException("Client mode " + clientMode + " in the MQTT
message is not expected");
}

final JSONArray servicesArray = json.getJSONArray("services");
if (servicesArray.length() > 1) {
    throw new RuntimeException("Services in the MQTT message has more than 1
service");
}
final String service = servicesArray.get(0).toString();
if (!service.equals("SSH")) {
    throw new RuntimeException("Service " + service + " is not supported");
}

// Start the destination local proxy in a separate process to connect to the
SSH Daemon listening port 22
final ProcessBuilder pb = new ProcessBuilder("localproxy",
    "-t", accessToken,
    "-r", region,
    "-d", "localhost:22");
pb.start();
}
catch (Exception e) {
    log.error("Failed to start the local proxy", e);
}
}
}
```

## Configuring a Remote Device

If you want to deliver the destination client access token to the remote device through methods other than subscribing to the reserved IoT MQTT topic, you might need two components on the remote device:

- A destination client access token listener.
- A local proxy.

The destination client access token listener should work with the client access token delivery mechanism of your choice. It must be able to start a local proxy in destination mode.

# Device Provisioning

When you provision a device with AWS IoT, you must create resources so your devices and AWS IoT can communicate securely. Other resources can be created to help you manage your device fleet. The following resources can be created during the provisioning process:

- An IoT thing.

IoT things are entries in the AWS IoT device registry. Each thing has a unique name and set of attributes, and is associated with a physical device. Things can be defined using a thing type or grouped into thing groups. For more information, see [Managing Devices with AWS IoT \(p. 93\)](#).

Although not required, creating a thing makes it possible to manage your device fleet more effectively by searching for devices by thing type, thing group, and thing attributes. For more information, see [Fleet Indexing Service \(p. 514\)](#).

- An X.509 certificate.

Devices use X.509 certificates to perform mutual authentication with AWS IoT. You can register an existing certificate or have AWS IoT generate and register a new certificate for you. You associate a certificate with a device by attaching it to the thing that represents the device. You must also copy the certificate and associated private key onto the device. Devices present the certificate when connecting to AWS IoT. For more information, see [Authentication \(p. 120\)](#).

- An IoT policy.

IoT policies define the operations a device can perform in AWS IoT. IoT policies are attached to device certificates. When a device presents the certificate to AWS IoT, it is granted the permissions specified in the policy. For more information, see [Authorization \(p. 137\)](#). Each device needs a certificate to communicate with AWS IoT.

AWS IoT supports automated fleet provisioning using provisioning templates. Provisioning templates describe the resources AWS IoT requires to provision your device. Templates contain variables that enable you to use one template to provision multiple devices. When you provision a device, you specify values for the variables specific to the device using a dictionary or *map*. To provision another device, specify new values in the dictionary.

You can use automated provisioning whether or not your devices have unique certificates (and their associated private key).

## Provisioning Devices That Don't Have Device Certificates Using Fleet Provisioning

The fleet provisioning feature is in beta and is subject to change. We recommend that you use this feature with test devices only.

When you use AWS IoT fleet provisioning, AWS IoT can generate and securely deliver device certificates and private keys to your devices when they connect to AWS IoT for the first time. Device certificates are registered for day-to-day use. There are two ways to use fleet provisioning:

- By claim.
- By trusted user.

## Provisioning by Claim

Devices can be manufactured with a provisioning claim certificate and private key (which are special purpose credentials) embedded in them. If these certificates are registered with AWS IoT, the service can exchange them for unique device certificates that the device can use for regular operations. This process includes the following steps:

1. You use the `CreateProvisioningTemplate` API to create a provisioning template. This API returns a template ARN. For more information, see [Fleet Provisioning APIs \(p. 498\)](#). You can also create a fleet provisioning template in the AWS IoT console.
  - a. From the navigation pane, choose **Onboard**, and then choose **Fleet provisioning templates**.
  - b. Choose **Create** and follow the prompts.
2. You create certificates and associated private keys to be used as provisioning claims.
3. You register these certificates with AWS IoT and associate an IoT policy that restricts the use of the certificates. The following example IoT policy restricts the use of the certificate associated with this policy to provisioning devices.

```
{
    "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/$aws/certificates/create/*",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/provisioning-
templates/templateName/provision/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/certificates/
create/*",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/provisioning-
templates/templateName/provision/*"
            ]
        }
    ]
}
```

4. You give the AWS IoT service permission to create or update IoT resources such as things and certificates in your account when provisioning devices. You do this by attaching the `AWSIoTThingsRegistration` managed policy to an IAM role (called the provisioning role) that trusts the AWS IoT service principal.
5. The device is manufactured with the provisioning claim securely embedded in it.
6. The device uses the provisioning claim to authenticate with AWS IoT using the AWS IoT Device SDK.

7. The device receives a device certificate and private key that it securely stores and uses for future authentication with AWS IoT. At the same time, the Fleet Provisioning service creates cloud resources such as IoT things, thing groups, and attributes, as defined in the provisioning template.

**Important**

Provisioning claim private keys should be secured at all times, including on the device. We recommend that you use AWS IoT CloudWatch metrics and logs to monitor for indications of misuse. If you detect misuse, disable the provisioning claim certificate so it cannot be used for device provisioning.

## Provisioning by Trusted User

In many cases, a device connects to AWS IoT for the first time when an end user or installation technician uses a mobile app to configure the device in its deployed location. This process includes the following steps:

1. You use the `CreateProvisioningTemplate` API to create a provisioning template. This API returns a template ARN.
2. You create an IAM role that is used by a trusted user to initiate the provisioning process. The provisioning template allows only that user to provision a device. For example:

```
{  
    "Effect": "Allow",  
    "Action": [  
        "iot:CreateProvisioningClaim",  
    ],  
    "Resource": [  
        "arn:aws:iot:us-west-2:123456789012:provisioningtemplate/templateName"  
    ]  
}
```

3. You give the AWS IoT service permission to create or update IoT resources, such as things and certificates in your account when provisioning devices. You do this by attaching the `AWSIoTThingsRegistration` managed policy to an IAM role (called the *provisioning role*) that trusts the AWS IoT service principal.
4. The trusted user signs in to your provisioning mobile app or web service.
5. The mobile app or web application uses the IAM role and the `CreateProvisioningClaim` API to obtain a temporary provisioning claim from AWS IoT.
6. The mobile app or web application supplies the provisioning claim to the device along with configuration information, such as Wi-Fi credentials.
7. The device uses the temporary provisioning claim to connect to AWS IoT using the [AWS IoT Device and Mobile SDKs \(p. 706\)](#).
8. The device receives a unique device certificate and private key that it securely stores and uses for future connections to AWS IoT. At the same time, the Fleet Provisioning service creates cloud resources such as IoT things, thing groups, and attributes, as defined in the provisioning template.

## Fleet Provisioning APIs

There are three categories of API used in fleet provisioning:

- Control plane API used to create and manage fleet provisioning templates and to configure trusted user policies.
  - [CreateProvisioningTemplate](#)
  - [CreateProvisioningTemplateVersion](#)

- [DeleteProvisioningTemplate](#)
- [DeleteProvisioningTemplateVersion](#)
- [DescribeProvisioningTemplate](#)
- [DescribeProvisioningTemplateVersion](#)
- [ListProvisioningTemplates](#)
- [ListProvisioningTemplateVersions](#)
- Control plane API are used by a trusted user to generate a temporary onboarding claim which is passed to the device during Wi-Fi config or similar method. There is one API in this category: [CreateProvisioningClaim](#).
- API used by devices during the provisioning process (using a provisioning claim embedded in a device or one passed to it by a trusted user).

## Device Provisioning MQTT API

The Fleet Provisioning service supports two MQTT APIs: [CreateKeysAndCertificate](#) and [RegisterThing](#).

### CreateKeysAndCertificate

Examples in this document use JSON for readability, but you can also use the Concise Binary Object Representation (CBOR) format. The new certificate will have the `PENDING_ACTIVATION` status. When you use the [RegisterThing](#) API to provision a thing, the certificate status changes to `ACTIVE` or `INACTIVE` based on the template definition.

To create a certificate, publish a message on `$aws/certificates/create/cbor` or `$aws/certificates/create/json`.

Request payload:

```
{}
```

Response payload:

To receive the response, subscribe to `$aws/certificates/create/cbor/accepted` or `$aws/certificates/create/json/accepted`. For more information about connecting to the message broker and using AWS IoT reserved topics, see [Lifecycle Events \(p. 693\)](#) and [Reserved Topics \(p. 245\)](#).

```
{
    "certificateId": "string",
    "certificatePem": "string",
    "privateKey": "string",
    "certificateOwnershipToken": "string"
}
```

`certificateId`

The certificate ID.

`certificatePem`

The certificate data, in PEM format.

`privateKey`

The private key.

#### certificateOwnershipToken

The token to prove ownership of the certificate during provisioning.

To receive error responses, subscribe to `$aws/certificates/create/cbor/rejected` or `$aws/certificates/create/json/rejected`.

Error payload:

```
{  
    "statusCode": int,  
    "errorCode": "string",  
    "errorMessage": "string"  
}
```

#### statusCode

The status code.

#### errorCode

The error code.

#### errorMessage

The error message.

For more information, see [CreateKeysAndCertificate](#) in the AWS IoT API Reference.

## RegisterThing

To provision a thing, publish a message on `$aws/provisioning-templates/templateName/provision/cbor` or `$aws/provisioning-templates/templateName/provision/json`.

Request payload:

```
{  
    "certificateOwnershipToken": "string",  
    "parameters": {  
        "string": "string"  
    },  
}
```

#### templateName

The provisioning template name.

#### certificateOwnershipToken

The token to prove ownership of the certificate. The token is generated by AWS IoT when you create a certificate over MQTT.

#### parameters

Optional. Name-value pairs to send to devices during provisioning.

To receive the response, subscribe to `$aws/provisioning-templates/templateName/provision/cbor/accepted` or `$aws/provisioning-templates/templateName/provision/json/accepted`.

Response payload:

```
{  
    "deviceConfiguration": {  
        "string": "string"  
    },  
    "thingName": "string"  
}
```

**thingName**

The name of the IoT thing created during provisioning.

**deviceConfiguration**

The device configuration defined in the template.

To receive error responses, subscribe to `$aws/provisioning-templates/templateName/provision/cbor/rejected` or `$aws/provisioning-templates/templateName/provision/json/rejected`.

**Error payload:**

```
{  
    "statusCode": int,  
    "errorCode": "string",  
    "errorMessage": "string"  
}
```

**statusCode**

The status code.

**errorCode**

The error code.

**errorMessage**

The error message.

For more information, see [RegisterThing](#) in the AWS IoT API Reference.

## Provisioning Devices That Have Device Certificates

The fleet provisioning feature is in beta and is subject to change. We recommend that you use this feature with test devices only.

AWS IoT provides three ways to provision devices when they already have a device certificate (and associated private key) on them:

- Single-thing provisioning with a provisioning template. This is a good option if you only need to provision devices one at a time.
- Just-in-time provisioning (JITP) with a template that provisions a device when it first connects to AWS IoT. This is a good option if you need to register large numbers of devices, but you don't have information about them that you can assemble into a bulk provisioning list.

- Bulk registration. This option allows you to specify a list of single-thing provisioning template values that are stored in a file in an S3 bucket. This approach works well if you have a large number of known devices whose desired characteristics you can assemble into a list.

## Single Thing 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 [Provisioning Templates \(p. 505\)](#).

[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/cd82bb924d4c6ccbb14986dcba4f40f30d892cc6b3ce7ad5008ed6542eea2b049",  
        "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. To provision the device, you must enable automatic registration and associate a provisioning template with the CA certificate used to sign the device certificate.

You can make these settings when you register a 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 file://<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 file://<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 uses it to call [RegisterThing](#). The JITP workflow first registers a certificate with a status value of PENDING\_ACTIVATION. When the device provisioning flow is complete, the status of the certificate is changed to ACTIVE.

AWS IoT defines the following parameters that you can declare and reference in 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 limited to what JITP can extract from the subject field of the certificate of the device being provisioned. The AWS::IoT::Certificate::Id parameter refers to an internally generated ID, not an ID that is contained in the certificate. You can get the value of this ID using the `principal()` function inside an AWS IoT rule.

The following JSON file is an example of a complete JITP template. The value of the `templateBody` field must be a JSON object specified as an escaped string and can use only the values in the preceding list. You can use a variety of tools to create the required JSON output, such as `json.dumps` (Python) or `JSON.stringify` (Node). The value of the `roleARN` field must be the ARN of a role that has the `AWSIoTThingsRegistration` attached to it. Also, your template can use an existing `PolicyName` instead of the inline `PolicyDocument` in the example. (The first example adds line breaks for readability, but you can copy and paste the template that appears directly below it.)

```
{
    "templateBody" : "{
        \r\n        \"Parameters\" : { \r\n            \r\n                \"AWS::IoT::Certificate::CommonName\": { \r\n                    \r\n                    \"Type\": \"String\"\r\n                }, \r\n                \r\n                \"AWS::IoT::Certificate::SerialNumber\": { \r\n                    \r\n                    \"Type\": \"String\"\r\n                }, \r\n                \r\n                \"AWS::IoT::Certificate::Country\": { \r\n                    \r\n                    \"Type\": \"String\"\r\n                }, \r\n                \r\n                \"AWS::IoT::Certificate::Id\": { \r\n                    \r\n                    \"Type\": \"String\"\r\n                }\r\n            }, \r\n            \r\n            \"Resources\": { \r\n                \r\n                \"thing\": { \r\n                    \r\n                    \"Type\": \"AWS::IoT::Thing\", \r\n                    \r\n                    \"Properties\": { \r\n                        \r\n                        \"ThingName\": { \r\n                            \r\n                            \"Ref\": \"AWS::IoT::Certificate::CommonName\"\r\n                        }, \r\n                        \r\n                        \"AttributePayload\": { \r\n                            \r\n                            \"version\": \"v1\", \r\n                            \r\n                            \"serialNumber\": { \r\n                                \r\n                                \"Ref\": \"AWS::IoT::Certificate::SerialNumber\"\r\n                            }\r\n                        }\r\n                    }\r\n                }\r\n            }\r\n        }\r\n    }"
}
```

```

        \\"OverrideSettings\": {\r\n
            \\"AttributePayload\": \"MERGE\", \r\n
            \\"ThingTypeNames\": \"REPLACE\", \r\n
            \\"ThingGroups\": \"DO NOTHING\"\r\n } \r\n }, \r\n
        \\"certificate\": {\r\n
            \\"Type\": \"AWS::IoT::Certificate\", \r\n
            \\"Properties\": {\r\n
                \\"CertificateId\": {\r\n        \\"Ref\": \"AWS::IoT::Certificate::Id\" \r\n    }, \r\n
                \\"Status\": \"ACTIVE\" \r\n            }, \r\n
                \\"OverrideSettings\": {\r\n
                    \\"Status\": \"DO NOTHING\" \r\n            } \r\n        }, \r\n
            \\"policy\": {\r\n
                \\"Type\": \"AWS::IoT::Policy\", \r\n
                \\"Properties\": {\r\n
                    \\"PolicyDocument\": \"{
                        \\"Version\": \"2012-10-17\",
                        \\"Statement\": [
                            {
                                \\"Effect\": \"Allow\",
                                \\"Action\": [\"iot:Publish\"],
                                \\"Resource\": [\"arn:aws:iot:us-east-1:123456789012:topic/sample/topic\"] }
                        ] \r\n                } \r\n            } \r\n        },
        \\"roleArn\" : \"arn:aws:iam::123456789012:role/Provisioning-JITP\" \r\n
    }
}

```

Here is a version you can copy and paste:

```
{
    "templateBody" : "{\r\n    \\"Parameters\": {\r\n        \\"AWS::IoT::Certificate::CommonName\": {\r\n            \\"Type\": \"String\" \r\n        }, \r\n        \\"AWS::IoT::Certificate::SerialNumber\": {\r\n            \\"Type\": \"String\" \r\n        }, \r\n        \\"AWS::IoT::Certificate::Country\": {\r\n            \\"Type\": \"String\" \r\n        }, \r\n        \\"AWS::IoT::Certificate::Id\": {\r\n            \\"Type\": \"String\" \r\n        }, \r\n        \\"Resources\": {\r\n            \\"thing\": {\r\n                \\"Properties\": {\r\n                    \\"Type\": \"AWS::IoT::Thing\", \r\n                    \\"ThingName\": {\r\n                        \\"Ref\": \"AWS::IoT::Certificate::CommonName\" \r\n                    }, \r\n                    \\"Attributes\": {\r\n                        \\"serialNumber\": {\r\n                            \\"Ref\": \"AWS::IoT::Certificate::SerialNumber\" \r\n                        }, \r\n                        \\"ThingTypeNames\": \"lightBulb-versionA\", \r\n                        \\"ThingGroups\": [\"v1-lightbulbs\"], \r\n                        \\"OverrideSettings\": {\r\n                            \\"AttributePayload\": \"MERGE\", \r\n                            \\"ThingGroups\": \"DO NOTHING\" \r\n                        }, \r\n                        \\"certificate\": {\r\n                            \\"Type\": \"AWS::IoT::Certificate\", \r\n                            \\"Properties\": {\r\n                                \\"CertificateId\": {\r\n                                    \\"Ref\": \"AWS::IoT::Certificate::Id\" \r\n                                }, \r\n                                \\"Status\": \"ACTIVE\" \r\n                            }, \r\n                            \\"OverrideSettings\": {\r\n                                \\"Status\": \"DO NOTHING\" \r\n                            }, \r\n                            \\"policy\": {\r\n                                \\"Type\": \"AWS::IoT::Policy\", \r\n                                \\"Properties\": {\r\n                                    \\"PolicyDocument\": \"{
  \\"Version\": \"2012-10-17\",
  \\"Statement\": [
  {
  \\"Effect\": \"Allow\",
  \\"Action\": [\"iot:Publish\"],
  \\"Resource\": [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }
  ] \r\n                                    } \r\n                                } \r\n                            },
                            \\"roleArn\" : \"arn:aws:iam::123456789012:role/JITPRole\" \r\n
                        }
}
}
```

This sample template declares values for the `AWS::IoT::Certificate::CommonName`, `AWS::IoT::Certificate::SerialNumber`, `AWS::IoT::Certificate::Country`, and

`AWS::IoT::Certificate::Id` provisioning parameters that are extracted from the certificate and used in the Resources section. The JITP workflow then uses this template to perform the following actions:

- Register a certificate and set its status to PENDING\_ACTIVE.
- Create one thing resource.
- Create one policy resource.
- Attach the policy to the certificate.
- Attach the certificate to the thing.
- Update the certificate status to ACTIVE.

You should be able to see the certificate registration as a logged event (`RegisterCACertificate`) in AWS CloudTrail. You can also use CloudTrail to troubleshoot issues with your JITP template.

**Note**

JITP calls other AWS IoT control plane APIs during the provisioning process. These calls might exceed the [AWS IoT Throttling Quotas](#) set for your account and result in throttled calls. Contact [AWS Customer Support](#) to raise your throttling quotas, if necessary.

## Bulk Registration

You can use the `start-thing-registration-task` command to register things in bulk. This command takes a registration template, an S3 bucket name, a key name, and a role ARN that allows access to the file in the S3 bucket. The file in the 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 registering a single device. For example:

```
{"ThingName": "foo", "SerialNumber": "123", "CSR": "csr1"}  
{"ThingName": "bar", "SerialNumber": "456", "CSR": "csr2"}
```

The following bulk registration-related APIs might be useful:

- [ListThingRegistrationTasks](#): Lists the current bulk thing provisioning tasks.
- [DescribeThingRegistrationTask](#): Provides information about a specific bulk thing registration task.
- [StopThingRegistrationTask](#): Stops a bulk thing registration task.
- [ListThingRegistrationTaskReports](#): Used to check the results and failures for a bulk thing registration task.

**Note**

- Only one bulk registration operation task can run at a time (per account).
- Bulk registration operations call other AWS IoT control plane APIs. These calls might exceed the [AWS IoT Throttling Quotas](#) in your account and cause throttle errors. Contact [AWS Customer Support](#) to raise your AWS IoT throttling quotas, if necessary.

## Provisioning Templates

A provisioning 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`.

There are two types of provisioning templates in AWS IoT. One is used for just-in-time provisioning (JITP) and bulk registration and the second is used for fleet provisioning.

## Parameters Section

The **Parameters** section declares the parameters used in 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 IoT 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`: Optional. A list of name-value pairs.
- `ThingTypeName`: Optional. String for an associated thing type for the thing.
- `ThingGroups`: Optional. A list of groups to which the thing belongs.

## Certificate Resources

You can specify certificates in one of the following ways:

- A certificate signing request (CSR).
- A certificate ID of an existing device certificate. (Only certificate IDs can be used with a fleet provisioning template.)
- 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 these methods. For example, if you use a CSR, you cannot also specify a certificate ID or a device certificate. For more information, see [AWS IoT and Certificates](#).

For more information, see [X.509 Certificate Overview \(p. 120\)](#).

Certificate resources are declared using the following properties:

- `CertificateSigningRequest`: String.
- `CertificateID`: String.
- `CertificatePem`: String.
- `CACertificatePem`: String.
- `Status`: Optional. String that can be ACTIVE or INACTIVE. Defaults to ACTIVE.

Examples:

- Certificate specified with a CSR:

```
{  
    "certificate" : {  
        "Type" : "AWS::IoT::Certificate",  
        "Properties" : {  
            "CertificateSigningRequest": {"Ref" : "CSR"},  
            "Status" : "ACTIVE"  
        }  
    }  
}
```

- Certificate specified with an existing certificate ID:

```
{  
    "certificate" : {  
        "Type" : "AWS::IoT::Certificate",  
        "Properties" : {  
            "CertificateId": {"Ref" : "CertificateId"}  
        }  
    }  
}
```

- Certificate specified with an existing certificate .pem and CA certificate .pem:

```
{  
    "certificate" : {  
        "Type" : "AWS::IoT::Certificate"
```

```
    "Properties" : {
        "CACertificatePem": {"Ref" : "CACertificatePem"},  
        "CertificatePem": {"Ref" : "CertificatePem"}  
    }
}
```

## Policy Resources

Policy resources are declared using one of the following properties:

- **PolicyName**: Optional. String. Defaults to a hash of the policy document. If you are using an existing AWS IoT policy, for the `PolicyName` property, enter the name of the policy. Do not include the `PolicyDocument` property.
- **PolicyDocument**: Optional. A JSON object specified as an escaped string. 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`.

**MERGE**

Valid only for the `ThingGroups` and `AttributePayload` properties of a thing. Merge the existing attributes or group memberships of the thing with those specified in the template.

When you declare a thing resource, you can specify `OverrideSettings` for the following properties:

- `ATTRIBUTE_PAYLOAD`
- `THING_TYPE_NAME`
- `THING_GROUPS`

When you declare a certificate resource, you can specify `OverrideSettings` for the `Status` property.

`OverrideSettings` are not available for policy resources.

## Resource Example

The following template snippet declares a thing, a certificate, and a policy:

```
{
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "AttributePayload" : { "version" : "v1", "serialNumber" : {"Ref" : "SerialNumber"} },
                "ThingTypeName" : "lightBulb-versionA",
                "ThingGroups" : [ "v1-lightbulbs", {"Ref" : "Location"} ]
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "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 belongs.

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 for JITP and Bulk Registration

The following JSON file is an example of a complete provisioning template that specifies the certificate with a CSR:

(The `PolicyDocument` field value must be a JSON object specified as an escaped string.)

```
{
    "Parameters" : {
        "SerialNumber" : {
            "Type" : "String"
        },
        "Location" : {
            "Type" : "String",
            "Default" : "WA"
        },
        "CSR" : {
            "Type" : "String"
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "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 following JSON file is an example of a complete provisioning template that specifies an existing certificate with a certificate ID:

```
{
    "Parameters" : {
        "SerialNumber" : {
            "Type" : "String"
        },
        "Location" : {
            "Type" : "String",
            "Default" : "WA"
        }
    }
}
```

```

        },
        "CertificateId" : {
            "Type" : "String"
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "AttributePayload" : { "version" : "v1", "serialNumber" : {"Ref" : "SerialNumber"} },
                "ThingTypeName" : "lightBulb-versionA",
                "ThingGroups" : ["v1-lightbulbs", {"Ref" : "Location"}]
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateId": {"Ref" : "CertificateId"}
            }
        },
        "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\"] }] }"
            }
        }
    }
}

```

## Fleet Provisioning

Fleet provisioning templates are used by AWS IoT to set up cloud and device configuration. These templates use the same parameters and resources as the JITP and bulk registration templates. For more information, see [Provisioning Templates \(p. 505\)](#). Fleet provisioning templates can contain a **Mappings** section and a **DeviceConfiguration** section. You can use intrinsic functions inside a fleet provisioning template to generate device specific configuration. Fleet provisioning templates are named resources and are identified by ARNs (for example, `arn:aws:iot:us-west-2:1234568788:provisioningtemplate/templateName`).

## Mappings

The optional **Mappings** section matches a key to a corresponding set of named values. For example, if you want to set values based on an AWS Region, you can create a mapping that uses the AWS Region name as a key and contains the values you want to specify for each specific region. You use the `Fn::FindInMap` intrinsic function to retrieve values in a map.

You cannot include parameters, pseudo parameters, or call intrinsic functions in the **Mappings** section.

## Device Configuration

The device configuration section contains arbitrary data you want to send to your devices when provisioning. For example:

```
{
    "DeviceConfiguration": {
        "Foo": "Bar"
    }
}
```

}

## Intrinsic Functions

Intrinsic functions are used in any section of the provisioning template except the `Mappings` section.

### `Fn::Join`

Appends a set of values into a single value, separated by the specified delimiter. If a delimiter is the empty string, the set of values are concatenated with no delimiter.

### `Fn::Select`

Returns a single object from a list of objects by index.

#### **Important**

`Fn::Select` does not check for `null` values or if the index is out of bounds of the array. Both conditions result in a provisioning error, so you should ensure you chose a valid index value, and that the list contains non-null values.

### `Fn::FindInMap`

Returns the value corresponding to keys in a two-level map that is declared in the `Mappings` section.

### `Fn::Split`

Splits a string into a list of string values so you can select an element from the list of strings. You specify a delimiter that determine where the string is split (for example, a comma). After you split a string, use `Fn::Select` to select an element.

For example, if a comma-delimited string of subnet IDs is imported to your stack template, you can split the string at each comma. From the list of subnet IDs, use `Fn::Select` to specify a subnet ID for a resource.

### `Fn::Sub`

Substitutes variables in an input string with values that you specify. You can use this function to construct commands or outputs that include values that aren't available until you create or update a stack.

## Fleet Provisioning Template Example

```
{  
    "Parameters" : {  
        "SerialNumber": {  
            "Type": "String"  
        },  
        "DeviceLocation": {  
            "Type": "String"  
        }  
    },  
    "Mappings": {  
        "LocationTable": {  
            "Seattle": {  
                "LocationUrl": "https://example.aws"  
            }  
        }  
    },  
    "Resources" : {  
        "thing" : {  
            "Type" : "AWS::IoT::Thing",  
            "Properties": {  
                "ThingName": "MyThing",  
                "ThingType": "MyThingType",  
                "ThingDescription": "A description of my thing.",  
                "ThingARN": "  
                    Fn::GetAtt": "thing",  
                    "Arn"  
                "  
            }  
        }  
    }  
}
```

```

    "Properties" : {
        "AttributePayload" : {
            "version" : "v1",
            "serialNumber" : "serialNumber"
        },
        "ThingTypeName" : {"Fn::Join": ["", ["ThingPrefix_",
{"Ref":"SerialNumber"}]]},
        "ThingGroups" : ["v1-lightbulbs", "WA"],
        "BillingGroup": "LightBulbBillingGroup"
    },
    "OverrideSettings" : {
        "AttributePayload" : "MERGE",
        "ThingTypeName" : "REPLACE",
        "ThingGroups" : "DO NOTHING"
    }
},
"certificate" : {
    "Type" : "AWS::IoT::Certificate",
    "Properties" : {
        "CertificateId": {"Ref": "AWS::IoT::Certificate::Id"},
        "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"]
                }
            ]
        }
    }
},
"DeviceConfiguration": {
    "FallbackUrl": "https://www.example.com/test-site",
    "LocationUrl": {
        "Fn::FindInMap": ["LocationTable", {"Ref": "DeviceLocation"}, "LocationUrl"]
    }
}
}
}

```

# Fleet Indexing Service

Fleet Indexing is a managed service that you can use to index, search, and aggregate your registry data, shadow data, and device connectivity data (device lifecycle events) in the cloud. After you set up your fleet index, the service manages the indexing of updates for your thing groups, thing registries, and device shadows. For more information about aggregation queries, see [Querying for Aggregate Data \(p. 525\)](#). You can use a simple query language to search across this data. You can also create a [dynamic thing group](#) with a search query.

## Note

It might take about 30 seconds for the Fleet Indexing service to update the fleet index after a thing is created, updated, or deleted.

When you enable indexing, AWS IoT creates an index for your things or thing groups. After it's active, you can run queries on your index, such as finding all devices that are handheld and have more than 70 percent battery life. AWS IoT keeps the index continuously updated with your latest data.

`AWS_Things` is the index created for all of your things. `AWS_ThingGroups` is the index that contains all of your thing groups.

You can use the [AWS IoT console](#) to manage your indexing configuration and run your search queries. Choose the indexes you would like to use in the console settings page. If you prefer programmatic access, you can use the AWS SDKs or the AWS Command Line Interface (AWS CLI).

For information about pricing this and other services, see the [AWS IoT Device Management Pricing](#) page.

## Topics

- [Managing Thing Indexing \(p. 514\)](#)
- [Managing Thing Group Indexing \(p. 524\)](#)
- [Querying for Aggregate Data \(p. 525\)](#)
- [Query Syntax \(p. 530\)](#)
- [Example Thing Queries \(p. 531\)](#)
- [Example Thing Group Queries \(p. 532\)](#)

## Managing Thing Indexing

`AWS_Things` is the index created for all of your things. You can control what to index: registry data, shadow data, and device connectivity status data (driven by device lifecycle events).

## Enabling Thing Indexing

You use the `update-indexing-configuration` CLI command or the `UpdateIndexingConfiguration` API to create the `AWS_Things` index and control its configuration. The `--thing-indexing-configuration` (`thingIndexingConfiguration`) parameter allows you to control what kind of data (for example, registry, shadow, and device connectivity data) is indexed.

The `--thing-indexing-configuration` parameter takes a string with the following structure:

```
{  
    "thingIndexingMode": "OFF"|"REGISTRY"|"REGISTRY_AND_SHADOW",  
    "thingConnectivityIndexingMode": "OFF"|"STATUS",  
}
```

```

    "customFields": [
        { name: <field-name>, type: String | Number | Boolean },
        ...
    ]
}

```

The `thingIndexingMode` attribute controls what kind of data is indexed. Valid values are:

**OFF**

No indexing.

**REGISTRY**

Index registry data.

**REGISTRY\_AND\_SHADOW**

Index registry and thing shadow data.

The `thingConnectivityIndexingMode` attribute specifies if thing connectivity data is indexed. Valid values are:

**OFF**

Thing connectivity data is not indexed.

**STATUS**

Thing connectivity data is indexed.

The `customFields` attribute is a list of field and data type pairs. Aggregation queries can be performed over these fields based on the data type. The indexing mode you choose (REGISTRY or REGISTRY\_AND\_SHADOW) effects what fields can be specified in `customFields`. For example, if you specify the REGISTRY indexing mode, you cannot specify a field from a thing shadow. Custom fields must be specified in `customFields` to be indexed.

If there is a type inconsistency between a custom field in your configuration and the value being indexed, the Fleet Indexing service ignores the inconsistent value for aggregation queries. CloudWatch logs are helpful when troubleshooting aggregation query problems. For more information, see [Troubleshooting Aggregation Queries for the Fleet Indexing Service \(p. 712\)](#).

Managed fields contain data associated with IoT things, thing groups, and device shadows. The data type of managed fields are defined by AWS IoT. You specify the values of each managed field when you create an IoT thing. For example thing names, thing groups, and thing descriptions are all managed fields. The Fleet Indexing service indexes managed fields based on the indexing mode you specify:

- Managed fields for the registry

```

"managedFields" : [
    {name:thingId, type:String},
    {name:thingName, type:String},
    {name:registry.version, type:Number},
    {name:registry(thingType, type:String},
    {name:registry(thingGroupNames, type:String},
]

```

- Managed fields for thing shadows

```

"managedFields" : [
    {name:shadow.version, type:Number},
]

```

```
    {name:shadow.delta, type:Boolean}
]
```

- Managed fields for thing connectivity

```
"managedFields" : [
    {name:connectivity.timestamp, type:Number},
    {name:connectivity.version, type:Number},
    {name:connectivity.connected, type:Boolean}
]
```

- Managed fields for thing groups

```
"managedFields" : [
    {name:description, type:String},
    {name:parentGroupNames, type:String},
    {name:thingGroupId, type:String},
    {name:thingGroupName, type:String},
    {name:version, type:Number},
]
```

Managed fields cannot be changed or appear in `customFields`.

The following is an example of how to use `update-indexing-configuration` to configure indexing:

```
aws iot update-indexing-configuration --thing-indexing-configuration
'thingIndexingMode=REGISTRY_AND_SHADOW,customFields=[{name=attributes.version,type=Number},
{name=attributes.color,type=String},{name=shadow.desired.power,type=Boolean}]]
```

This command enables indexing for registry and shadow data. Aggregation queries work with the managed fields and the provided `customFields` based on the data type.

You can use the `get-indexing-configuration` CLI command or the [GetIndexingConfiguration](#) API to retrieve the current indexing configuration.

The following command shows how to use the `get-indexing-configuration` CLI command to retrieve the current thing indexing configuration where five custom fields (three registry custom fields and two shadow custom fields) are defined.

**aws iot get-indexing-configuration**

```
{
    "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "OFF"
    },
    "thingIndexingConfiguration": {
        "thingConnectivityIndexingMode": "STATUS",
        "customFields": [
            {
                "name": "attributes.customField_NUM",
                "type": "Number"
            },
            {
                "name": "shadow.desired.customField_STR",
                "type": "String"
            },
            {
                "name": "shadow.desired.customField_NUM",
                "type": "Number"
            }
        ]
}
```

```

        "name": "attributes.customField_STR",
        "type": "String"
    },
    {
        "name": "attributes.customField_BOOL",
        "type": "Boolean"
    }
],
"thingIndexingMode": "REGISTRY_AND_SHADOW",
"managedFields": [
    {
        "name": "shadow.hasDelta",
        "type": "Boolean"
    },
    {
        "name": "registry.thingGroupNames",
        "type": "String"
    },
    {
        "name": "connectivity.version",
        "type": "Number"
    },
    {
        "name": "registry.thingTypeName",
        "type": "String"
    },
    {
        "name": "connectivity.connected",
        "type": "Boolean"
    },
    {
        "name": "registry.version",
        "type": "Number"
    },
    {
        "name": "thingId",
        "type": "String"
    },
    {
        "name": "connectivity.timestamp",
        "type": "Number"
    },
    {
        "name": "thingName",
        "type": "String"
    },
    {
        "name": "shadow.version",
        "type": "Number"
    }
]
}
}

```

The following table provides the allowed combinations of `thingIndexingMode` and `thingConnectivityIndexingMode`, and their associated effects. The required `thingIndexingMode` parameter specifies if the AWS\_Things index contains just registry data or registry and shadow data. The optional `thingConnectivityIndexingMode` parameter specifies whether the index also contains connectivity status data (when devices have last connected and disconnected to AWS IoT).

| <code>thingIndexingMode</code> | <code>thingConnectivityIndexingMode</code> | Result                          |
|--------------------------------|--------------------------------------------|---------------------------------|
| OFF                            | <i>Not specified.</i>                      | No indexing or delete an index. |

| thingIndexingMode   | thingConnectivityIndexingMode | Result                                                                                                                                                          |
|---------------------|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OFF                 | OFF                           | Equivalent to the previous entry.                                                                                                                               |
| REGISTRY            | <i>Not specified.</i>         | Create or configure the AWS_Things index to index registry data only.                                                                                           |
| REGISTRY            | OFF                           | Equivalent to the previous entry.<br>(Only registry data is indexed.)                                                                                           |
| REGISTRY_AND_SHADOW | <i>Not specified.</i>         | Create or configure the AWS_Things index to index registry data and shadow data.                                                                                |
| REGISTRY_AND_SHADOW | OFF                           | Equivalent to the previous entry.<br>(Registry data and shadow data are indexed.)                                                                               |
| REGISTRY            | STATUS                        | Create or configure the AWS_Things index to index registry data and thing connectivity status data (REGISTRY_AND_CONNECTIVITY_STATUS).                          |
| REGISTRY_AND_SHADOW | STATUS                        | Create or configure the AWS_Things index to index registry data, shadow data, and thing connectivity status data (REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS). |

You can use the AWS IoT **update-indexing-configuration** CLI command to update your indexing configuration. The following examples show how to use the **update-indexing-configuration** CLI command.

Short syntax:

```
aws iot update-indexing-configuration --thing-indexing-configuration
"thingIndexingMode=REGISTRY_AND_SHADOW,thingConnectivityIndexingMode=STATUS,customFields=[{name=attributes.color,type=String},{name=shadow.desired.power,type=Boolean}]"
```

JSON syntax:

```
aws iot update-indexing-configuration --cli-input-json \'{ "thingIndexingConfiguration": {
    "thingIndexingMode": "REGISTRY_AND_SHADOW", "thingConnectivityIndexingMode": "STATUS",
    "customFields": [ { "name": "shadow.desired.power", "type": "Boolean" }, { "name": "attributes.color",
        "type": "String" }, { "name": "attributes.version", "type": "Number" } ] } }'
```

The output of these commands is:

```
{
  "thingIndexingConfiguration": {
    "thingConnectivityIndexingMode": "STATUS",
    "customFields": [
      {
        "type": "String",
        "name": "attributes.color"
      },
      {
        "type": "String",
        "name": "shadow.desired.power"
      }
    ]
  }
}
```

```

        "type": "Number",
        "name": "attributes.version"
    },
    {
        "type": "Boolean",
        "name": "shadow.desired.power"
    }
],
"thingIndexingMode": "REGISTRY_AND_SHADOW",
"managedFields": [
    {
        "type": "Boolean",
        "name": "connectivity.connected"
    },
    {
        "type": "String",
        "name": "registry.thingTypeName"
    },
    {
        "type": "String",
        "name": "thingName"
    },
    {
        "type": "Number",
        "name": "shadow.version"
    },
    {
        "type": "String",
        "name": "thingId"
    },
    {
        "type": "Boolean",
        "name": "shadow.hasDelta"
    },
    {
        "type": "Number",
        "name": "connectivity.timestamp"
    },
    {
        "type": "String",
        "name": "registry.thingGroupNames"
    },
    {
        "type": "Number",
        "name": "connectivity.version"
    },
    {
        "type": "Number",
        "name": "registry.version"
    }
]
},
"thingGroupIndexingConfiguration": {
    "thingGroupIndexingMode": "OFF"
}
}
}

```

In the following example, a new custom field is added to the configuration:

```
aws iot update-indexing-configuration --thing-indexing-configuration
'thingIndexingMode=REGISTRY_AND_SHADOW,customFields=[{name=attributes.version,type=Number},
{name=attributes.color,type=String},{name=shadow.desired.power,type=Boolean},
{name=shadow.desired.intensity,type=Number}]'
```

This command added `shadow.desired.intensity` to the indexing configuration.

**Note**

Updating the custom fields in indexing configuration overwrites all existing custom fields. Make sure to specify all custom fields when calling **update-indexing-configuration**.

After the index is rebuilt you can, use aggregation query on the newly added fields, search registry data, shadow data, and thing connectivity status data.

When changing the indexing mode, make sure all of your custom fields are valid using the new indexing mode. For example, if you start off with `REGISTRY_AND_SHADOW` mode with a custom field called `shadow.desired.temperature` you must delete the `shadow.desired.temperature` custom field before changing the indexing mode to `REGISTRY`. If your indexing configuration contains custom fields that are not indexed by the indexing mode, the update fails.

## Describing a Thing Index

The following command shows you how to use the **describe-index** CLI command to retrieve the current status of the thing index.

```
aws iot describe-index --index-name "AWS_Things"
{
    "indexName": "AWS_Things",
    "indexStatus": "BUILDING",
    "schema": "REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS"
}
```

The first time you enable indexing, AWS IoT builds your index. You can't query the index if `indexStatus` is in the `BUILDING` state. The `schema` for the things index indicates which type of data (`REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS`) is indexed.

Changing the configuration of your index causes the index to be rebuilt. During this process, the `indexStatus` is `REBUILDING`. You can execute queries on data in the things index while it is being rebuilt. 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. However, you can't 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 a Thing Index

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"
            },
            "connectivity": {
                "connected": false,
                "timestamp": "1556649874716"
            }
        },
        {
            "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
    },
    "connectivity": {
        "connected": true,
        "timestamp": 1556649855046
    }
},
"nextToken": "AQFCuvk7zZ3D9pOYMbFCeHbdZ+h=G"
}

```

In the JSON response, "connectivity" (as enabled by the `thingConnectivityIndexingMode=STATUS` setting) provides a Boolean value and a timestamp that indicates if the device is connected to AWS IoT Core. The device "mything1" disconnected (`false`) at POSIX time 1556649874716:

```

"connectivity": {
    "connected": false,
    "timestamp": 1556649874716
}

```

```
}
```

The device "mything2" connected (true) at POSIX time 1556649855046:

```
"connectivity": {
    "connected":true,
    "timestamp":1556649855046
}
```

Timestamps are given in milliseconds since epoch, so 1556649855046 represents 6:44:15.046 PM on Tuesday, April 30, 2019 (GMT).

#### Important

If a device has been disconnected for approximately an hour, the "timestamp" value of the connectivity status might be missing.

## Restrictions and Limitations

These are the restrictions and limitations for AWS\_Things.

### Shadow fields with complex types

A shadow field is indexed only if the value of the field is a simple type, a JSON object that does not contain an array, or an array that consists entirely of simple types. Simple type means a string, number, or one of the literals true or false. For example, given the following shadow state, the value of field "palette" is not indexed because it's an array that contains items of complex types. The value of field "colors" is indexed because each value in the array is a string.

```
{
  "state": {
    "reported": {
      "switched": "ON",
      "colors": [ "RED", "GREEN", "BLUE" ],
      "palette": [
        {
          "name": "RED",
          "intensity": 124
        },
        {
          "name": "GREEN",
          "intensity": 68
        },
        {
          "name": "BLUE",
          "intensity": 201
        }
      ]
    }
  }
}
```

### Nested shadow field names

The names of nested shadow fields are stored as a period (.) delimited string. For example, given a shadow document:

```
{
  "state": {
    "desired": {
      "one": {
        "two": {

```

```

        "three": "v2"
    }
}
}
}
}
```

the name of field `three` is stored as `desired.one.two.three`. If you also have a shadow document like this:

```
{
  "state": {
    "desired": {
      "one.two.three": "v2"
    }
  }
}
```

both match a query for `shadow.desired.one.two.three:v2`. As a best practice, do not use periods in shadow field names.

#### Shadow metadata

A field in a shadow's metadata section is indexed, but only if the corresponding field in the shadow's `"state"` section is indexed. (In the previous example, the `"palette"` field in the shadow's metadata section is not indexed either.)

#### Unregistered shadows

If you use [UpdateThingShadow](#) to create a shadow using a thing name that hasn't been registered in your AWS IoT account, fields in this shadow are not indexed.

#### Numeric values

If any registry or shadow data is recognized by the service as a numeric value, it's indexed as such. You can form queries involving ranges and comparison operators on numeric values (for example, `"attribute.foo<5"` or `"shadow.reported.foo:[75 TO 80]"`). To be recognized as numeric, the value of the data must be a valid JSON number type literal (an integer in the range  $-2^{53}...2^{53}-1$  or a double-precision floating point with optional exponential notation) or part of an array that contains only such values.

#### Null values

Null values are not indexed.

#### Maximum number of custom fields for aggregation queries

5

Maximum number of requested percentiles for aggregation queries.

100

## Authorization

You can specify the things index as a resource ARN in an AWS IoT policy action, as follows.

| Action                       | Resource                                                                                                                |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| <code>iot:SearchIndex</code> | An index ARN (for example, <code>arn:aws:iot:&lt;your-aws-region&gt;&lt;your-aws-account&gt;:index/AWS_Things</code> ). |

| Action            | Resource                                                                    |
|-------------------|-----------------------------------------------------------------------------|
| iot:DescribeIndex | An index ARN (for example, arn:aws:iot:<your-aws-region>:index/AWS_Things). |

**Note**

If you have permissions to query the fleet index, you can access the data of things across the entire fleet.

## Managing Thing Group Indexing

AWS\_ThingGroups is the index that contains all of your thing groups. You can use this index to search for groups based on group name, description, attributes, and all parent group names.

### Enabling Thing Group Indexing

You can use the `thing-group-indexing-configuration` setting in the [UpdateIndexingConfiguration](#) API to create the AWS\_ThingGroups index and control its configuration. You can use the [GetIndexingConfiguration](#) API to retrieve the current indexing configuration.

Use the **get-indexing-configuration** CLI command to retrieve the current thing and thing group indexing configurations.

**aws iot get-indexing-configuration**

```
{
    "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "ON"
    }
}
```

Use the **update-indexing-configuration** CLI command to update the thing group indexing configurations.

```
aws iot update-indexing-configuration --thing-group-indexing-configuration
    thingGroupIndexingMode=ON
```

**Note**

You can also update configurations for both thing and thing group indexing in a single command, as follows.

```
aws iot update-indexing-configuration --thing-indexing-configuration
    thingIndexingMode=REGISTRY --thing-group-indexing-configuration
    thingGroupIndexingMode=ON
```

The following are valid values for `thingGroupIndexingMode`.

OFF

No indexing/delete index.

ON

Create or configure the AWS\_ThingGroups index.

## Describing Group Indexes

Use the **describe-index** CLI command to retrieve the current status of the AWS\_ThingGroups index.

```
aws iot describe-index --index-name "AWS_ThingGroups"
{
    "indexStatus": "ACTIVE",
    "indexName": "AWS_ThingGroups",
    "schema": "THING_GROUPS"
}
```

AWS IoT builds your index the first time you enable indexing. You can't query the index if the indexStatus is BUILDING.

## Querying a Thing Group Index

Use the **search-index** CLI command to query data in the index:

```
aws iot search-index --index-name "AWS_ThingGroups" --query-string
"thingGroupName:mythinggroup*"
```

## Authorization

You can specify the thing groups index as a resource ARN in an AWS IoT policy action, as follows.

| Action            | Resource                                                                         |
|-------------------|----------------------------------------------------------------------------------|
| iot:SearchIndex   | An index ARN (for example, arn:aws:iot:<your-aws-region>:index/AWS_ThingGroups). |
| iot:DescribeIndex | An index ARN (for example, arn:aws:iot:<your-aws-region>:index/AWS_ThingGroups). |

## Querying for Aggregate Data

AWS IoT provides three APIs (`GetStatistics`, `GetCardinality`, and `GetPercentiles`) that allow you to search your device fleet for aggregate data.

### GetStatistics

The `GetStatistics` API and the **get-statistics** CLI command return the count, average, sum, minimum, maximum, sum of squares, variance, and standard deviation for the specified aggregated field.

The **get-statistics** CLI command takes the following parameters:

`index-name`

The name of the index to search. The default value is AWS\_Things.

#### query-string

The query used to search the index. You can specify "\*" to get the count of all indexed things in your AWS account.

#### aggregationField

Optional. The field to aggregate. This field must be a managed or custom field defined when you call **update-indexing-configuration**. If you don't specify an aggregation field, `registry.version` is used as aggregation field.

#### query-version

The version of the query to use. The default value is 2017-09-30.

The type of aggregation field can affect the statistics returned.

## GetStatistics with String Values

If you aggregate on a string field, calling `GetStatistics` returns a count of devices that have attributes that match the query. For example:

```
aws iot get-statistics --aggregation-field 'attributes.stringAttribute' --query-string '*'
```

This command returns the number of devices that contain an attribute named `stringAttribute`:

```
{  
  "statistics": {  
    "count": 3  
  }  
}
```

## GetStatistics with Boolean Values

When you call `GetStatistics` with a boolean aggregation field:

- `AVERAGE` is the percentage of devices that match the query.
- `MINIMUM` is 0 or 1 according to the following rules:
  - If all the values for the aggregation field are `false`, `MINIMUM` is 0.
  - If all the values for the aggregation field are `true`, `MINIMUM` is 1.
  - If the values for the aggregation field are a mixture of `false` and `true`, `MINIMUM` is 0.
- `MAXIMUM` is 0 or 1 according to the following rules:
  - If all the values for the aggregation field are `false`, `MAXIMUM` is 0.
  - If all the values for the aggregation field are `true`, `MAXIMUM` is 1.
  - If the values for the aggregation field are a mixture of `false` and `true`, `MAXIMUM` is 1.
- `SUM` is the sum of the integer equivalent of the boolean values.
- `COUNT` is the number of things that match the query.

## GetStatistics with Numerical Values

When you call `GetStatistics` and specify an aggregation field of type `Number`, `GetStatistics` returns the following values:

#### count

The number of devices that have a field that matches the query.

average

The average of the numerical values that match the query.

sum

The sum of the numerical values that match the query.

minimum

The smallest of the numerical values that match the query.

maximum

The largest of the numerical values that match the query.

sumOfSquares

The sum of the squares of the numerical values that match the query.

variance

The variance of the numerical values that match the query. The variance of a set of values is the average of the squares of the differences of each value from the average value of the set.

stdDeviation

The standard deviation of the numerical values that match the query. The standard deviation of a set of values is a measure of how spread out the values are.

The following example shows how to call **get-statistics** with a numerical custom field.

```
aws iot get-statistics --aggregation-field 'attributes.numericAttribute2' --query-string '*'
```

```
{  
  "statistics": {  
    "count": 3,  
    "average": 33.33333333333336,  
    "sum": 100.0,  
    "minimum": -125.0,  
    "maximum": 150.0,  
    "sumOfSquares": 43750.0,  
    "variance": 13472.22222222222,  
    "stdDeviation": 116.06990230986766  
  }  
}
```

For numerical aggregation fields, if the field values exceed the maximum double value, the statistics values are empty

## GetCardinality

The [GetCardinality](#) API and the **get-cardinality** CLI command return the approximate count of unique values that match the query. For example, you might want to find the number of devices with battery levels at less than 50 percent:

```
aws iot get-cardinality --index-name AWS_Things --query-string "batterylevel > 50" --aggregation-field "shadow.reported.batterylevel".
```

This command returns the number of things with battery levels at more than 50 percent:

```
{
```

```
        "cardinality": 100
    }
```

cardinality is always returned by **get-cardinality** even if there are no matching fields. For example:

```
aws iot get-cardinality --query-string "thingName:Non-existent*" --aggregation-field
"attributes.customField_STR"
```

```
{
    "cardinality": 0
}
```

The **get-cardinality** CLI command takes the following parameters:

**index-name**

The name of the index to search. The default value is AWS\_Things.

**query-string**

The query used to search the index. You can specify "\*" to get the count of all indexed things in your AWS account.

**aggregationField**

The field to aggregate.

**query-version**

The version of the query to use. The default value is 2017-09-30.

## GetPercentiles

The [GetPercentiles](#) API and the **get-percentiles** CLI command groups the aggregated values that match the query into percentile groupings. The default percentile groupings are: 1,5,25,50,75,95,99, although you can specify your own when you call GetPercentiles. This function returns a value for each percentile group specified (or the default percentile groupings). The percentile group "1" contains the aggregated field value that occurs in approximately one percent of the values that match the query. The percentile group "5" contains the aggregated field value that occurs in approximately five percent of the values that match the query, and so on. The result is an approximation, the more values that match the query, the more accurate the percentile values.

The following example shows how to call the **get-percentiles** CLI command.

```
aws iot get-percentiles --query-string "thingName:\"" --aggregation-field
"attributes.customField_NUM" --percents 10 20 30 40 50 60 70 80 90 99
```

```
{
    "percentiles": [
        {
            "value": 3.0,
            "percent": 80.0
        },
        {
            "value": 2.5999999999999996,
            "percent": 70.0
        },
        {
            "value": 3.0,
            "percent": 90.0
        }
    ]
}
```

```
        },
        {
            "value": 2.0,
            "percent": 50.0
        },
        {
            "value": 2.0,
            "percent": 60.0
        },
        {
            "value": 1.0,
            "percent": 10.0
        },
        {
            "value": 2.0,
            "percent": 40.0
        },
        {
            "value": 1.0,
            "percent": 20.0
        },
        {
            "value": 1.4,
            "percent": 30.0
        },
        {
            "value": 3.0,
            "percent": 99.0
        }
    ]
}
```

The following command shows the output returned from **get-percentiles** when there are no matching documents.

```
aws iot get-percentiles --query-string "thingName:Non-existent*" --aggregation-field "attributes.customField_NUM"
```

```
{
    "percentiles": []
}
```

The **get-percentile** CLI command takes the following parameters:

**index-name**

The name of the index to search. The default value is AWS\_Things.

**query-string**

The query used to search the index. You can specify "\*" to get the count of all indexed things in your AWS account.

**aggregationField**

The field to aggregate, which must be of Number type.

**query-version**

The version of the query to use. The default value is 2017-09-30.

**percents**

Optional. You can use this parameter to specify custom percentile groupings.

## Authorization

You can specify the thing groups index as a resource ARN in an AWS IoT policy action, as follows.

| Action            | Resource                                                                                                                           |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------|
| iot:GetStatistics | An index ARN (for example, arn:aws:iot:<your-aws-region>:index/AWS_Things or arn:aws:iot:<your-aws-region>:index/AWS_ThingGroups). |

## Query Syntax

Queries are specified using a query syntax.

The query syntax supports the following features.

- Terms and phrases
- Searching fields
- Prefix search
- Range search
- Boolean operators AND, OR, NOT and -. The hyphen is used to exclude something from search results (for example, thingName:(tv\* AND -plasma)).
- Grouping
- Field grouping
- Escaping special characters (as with \)

The query syntax does not support the following features:

- Leading wildcard search (such as "\*xyz"), but searching for "\*" matches 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 isn't specified, AWS IoT searches for the term in all fields.
- All field names are case sensitive.
- Search is case insensitive. Words are separated by white space characters as defined by Java's `Character.isWhitespace(int)`.
- Indexing of device shadow data includes reported, desired, delta, and metadata sections.
- Device shadow and registry versions are not searchable, but are present in the response.
- The maximum number of terms in a query is 5.

# Example Thing Queries

Queries are specified in a query string using a 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 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", and so on.)                                                                                                                                       |
| thingTypeName:aa                                                                 | Queries for things that are associated with type aa.                                                                                                                                                                                                            |
| attributes.myAttribute:75                                                        | Queries for things with an attribute named "myAttribute" that has the value 75.                                                                                                                                                                                 |
| attributes.myAttribute:[75 TO 80]                                                | Queries for things with an attribute named "myAttribute" whose value falls within a numeric range (75–80, inclusive).                                                                                                                                           |
| attributes.myAttribute:{75 TO 80}                                                | Queries for things with an attribute named "myAttribute" whose value falls within the numeric range (>75 and <=80).                                                                                                                                             |
| attributes.serialNumber:["abcd" TO "abcf"]                                       | Queries for things with an attribute named "serialNumber" whose value is within an alphanumeric string range. This query returns things with a "serialNumber" attribute with values "abcd", "abce", or "abcf".                                                  |
| attributes.myAttribute:i*t                                                       | Queries for things with an attribute named "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 returns 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.                                                                                                                                                                                                            |
| NOT attributes.model:legacy                                                      | Queries for things where the attribute named "model" is not "legacy".                                                                                                                                                                                           |
| shadow.reported.stats.battery:{70 TO 100} (v2 OR v3) NOT attributes.model:legacy | Queries for things with the following: <ul style="list-style-type: none"> <li>The thing's shadow stats.battery attribute has a value between 70 and 100.</li> </ul>                                                                                             |

| Query String                                                                                            | Result                                                                                                                                                                                                                                                                                            |
|---------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                         | <ul style="list-style-type: none"> <li>The text "v2" or "v3" occurs in a thing's name, type name, or attribute values.</li> <li>The thing's <code>model</code> attribute is not set to "legacy".</li> </ul>                                                                                       |
| <code>shadow.reported.myvalues:2</code>                                                                 | Queries for things where the <code>myvalues</code> array in the shadow's reported section contains a value of 2.                                                                                                                                                                                  |
| <code>shadow.reported.location:* NOT shadow.desired.stats.battery:*</code>                              | Queries for things with the following: <ul style="list-style-type: none"> <li>The <code>location</code> attribute exists in the shadow's <code>reported</code> section.</li> <li>The <code>stats.battery</code> attribute does not exist in the shadow's <code>desired</code> section.</li> </ul> |
| <code>connectivity.connected:true</code>                                                                | Queries for all connected devices.                                                                                                                                                                                                                                                                |
| <code>connectivity.connected:false</code>                                                               | Queries for all disconnected devices.                                                                                                                                                                                                                                                             |
| <code>connectivity.connected:true AND connectivity.timestamp : [1557651600000 TO 1557867600000]</code>  | Queries for all connected devices with a connect timestamp $\geq$ 1557651600000 and $\leq$ 1557867600000. Timestamps are given in milliseconds since epoch.                                                                                                                                       |
| <code>connectivity.connected:false AND connectivity.timestamp : [1557651600000 TO 1557867600000]</code> | Queries for all disconnected devices with a disconnect timestamp $\geq$ 1557651600000 and $\leq$ 1557867600000. Timestamps are given in milliseconds since epoch.                                                                                                                                 |
| <code>connectivity.connected:true AND connectivity.timestamp &gt; 1557651600000</code>                  | Queries for all connected devices with a connect timestamp $>$ 1557651600000. Timestamps are given in milliseconds since epoch.                                                                                                                                                                   |
| <code>connectivity.connected:*</code>                                                                   | Queries for all devices with connectivity information present.                                                                                                                                                                                                                                    |

## Example Thing Group Queries

Queries are specified in a query string using a query syntax and passed to the [SearchIndex API](#). The following table lists some example query strings.

| Query String                                 | Result                                                                                                                          |
|----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| <code>abc</code>                             | Queries for "abc" in any field.                                                                                                 |
| <code>thingGroupName:myGroupThingName</code> | Queries for a thing group with name "myGroupThingName".                                                                         |
| <code>thingGroupName:my*</code>              | Queries for thing groups with names that begin with "my".                                                                       |
| <code>thingGroupName:ab?</code>              | Queries for thing groups with names that have "ab" plus one additional character (for example: "aba", "abb", "abc", and so on.) |

| Query String                                                        | Result                                                                                                                                                                                                                                                                      |
|---------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| attributes.myAttribute:75                                           | Queries for thing groups with an attribute named "myAttribute" that has the value 75.                                                                                                                                                                                       |
| attributes.myAttribute:[75 TO 80]                                   | Queries for thing groups with an attribute named "myAttribute" whose value falls within a numeric range (75–80, inclusive).                                                                                                                                                 |
| attributes.myAttribute:[75 TO 80]                                   | Queries for thing groups with an attribute named "myAttribute" whose value falls within the numeric range (>75 and <=80).                                                                                                                                                   |
| attributes.myAttribute:["abcd" TO "abcf"]                           | Queries for thing groups with an attribute named "myAttribute" whose value is within an alphanumeric string range. This query returns thing groups with a "serialNumber" attribute with values "abcd", "abce", or "abcf".                                                   |
| attributes.myAttribute:i*t                                          | Queries for thing groups with an attribute named "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 thing groups that combine terms using Boolean expressions. This query returns thing groups 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. |
| NOT attributes.myAttribute:cde                                      | Queries for thing groups where the attribute named "myAttribute" is not "cde".                                                                                                                                                                                              |
| parentGroupNames:(myParentThingGroupName)                           | Queries for thing groups whose parent group name matches "myParentThingGroupName".                                                                                                                                                                                          |
| parentGroupNames:(myParentThingGroupName OR myRootThingGroupName)   | Queries for thing groups whose parent group name matches "myParentThingGroupName" or "myRootThingGroupName".                                                                                                                                                                |
| parentGroupNames:(myParentThingGroupNa*)                            | Queries for thing groups whose parent group name begins with "myParentThingGroupNa".                                                                                                                                                                                        |

# AWS IoT Device Defender

AWS IoT Device Defender is a security service that allows you to audit the configuration of your devices, monitor connected devices to detect abnormal behavior, and mitigate security risks. It gives you the ability to enforce consistent security policies across your AWS IoT device fleet and respond quickly when devices are compromised.

IoT fleets can consist of large numbers of devices that have diverse capabilities, are long-lived, and are geographically distributed. These characteristics make fleet setup complex and error-prone. And because devices are often constrained in computational power, memory, and storage capabilities, this limits the use of encryption and other forms of security on the devices themselves. Also, devices often use software with known vulnerabilities. These factors make IoT fleets an attractive target for hackers and make it difficult to secure your device fleet on an ongoing basis.

AWS IoT Device Defender addresses these challenges by providing tools to identify security issues and deviations from best practices. AWS IoT Device Defender can audit device fleets to ensure they adhere to security best practices and detect abnormal behavior on devices.

## AWS Training and Certification

Take the following course to get started with AWS IoT Device Defender: [AWS IoT Device Defender Primer](#).

## Audit

An AWS IoT Device Defender audit looks at account- and device-related settings and policies to ensure security measures are in place. An audit can help you detect any drifts from security best practices or access policies (for example, multiple devices using the same identity, or overly permissive policies that allow one device to read and update data for many other devices). You can run audits as needed (*on-demand audits*) or schedule them to be run periodically (*scheduled audits*).

An AWS IoT Device Defender audit runs a set of predefined checks for common IoT security best practices and device vulnerabilities. Examples of predefined checks include policies that grant permission to read or update data on multiple devices, devices that share an identity (X.509 certificate), or certificates that are expiring or have been revoked but are still active.

## Issue Severity

Issue severity indicates the level of concern associated with each identified instance of noncompliance and the recommended time to remediation.

### Critical

Noncompliant audit checks with this severity identify issues that require urgent attention. Critical issues often allow bad actors with little sophistication and no insider knowledge or special credentials to easily gain access to or control of your assets.

### High

Noncompliant audit checks with this severity require urgent investigation and remediation planning after critical issues are addressed. Like critical issues, high severity issues often provide bad actors

with access to or control of your assets. However, high severity issues are often more difficult to exploit. They might require special tools, insider knowledge, or specific setups.

#### Medium

Noncompliant audit checks with this severity present issues that need attention as part of your continuous security posture maintenance. Medium severity issues might cause negative operational impact, such as unplanned outages due to malfunction of security controls. These issues might also provide bad actors with limited access to or control of your assets, or might facilitate parts of their malicious actions.

#### Low

Noncompliant audit checks with this severity often indicate security best practices were overlooked or bypassed. Although they might not cause an immediate security impact on their own, these lapses can be exploited by bad actors. Like medium severity issues, low severity issues require attention as part of your continuous security posture maintenance.

## Next Steps

To understand the types of audit checks that can be performed, see [Audit Checks \(p. 535\)](#). To learn how to perform an audit, see [How to Perform Audits \(p. 559\)](#). For information about service quotas that apply to audits, see [Service Quotas](#).

## Audit Checks

### Note

When you enable a check, data collection starts immediately. If there is a large amount of data in your account to collect, results of the check might not be available for some time after you enabled it.

The following audit checks are supported:

- [REVOKED\\_CA\\_CERT\\_CHECK \(p. 535\)](#)
- [DEVICE\\_CERTIFICATE\\_SHARED\\_CHECK \(p. 536\)](#)
- [DEVICE\\_CERTIFICATE\\_KEY\\_QUALITY\\_CHECK \(p. 537\)](#)
- [CA\\_CERTIFICATE\\_KEY\\_QUALITY\\_CHECK \(p. 538\)](#)
- [UNAUTHENTICATED\\_COGNITO\\_ROLE\\_OVERLY\\_PERMISSIVE\\_CHECK \(p. 539\)](#)
- [AUTENTICATED\\_COGNITO\\_ROLE\\_OVERLY\\_PERMISSIVE\\_CHECK \(p. 544\)](#)
- [IOT\\_POLICY\\_OVERLY\\_PERMISSIVE\\_CHECK \(p. 551\)](#)
- [IOT\\_ROLE\\_ALIAS\\_OVERLY\\_PERMISSIVE\\_CHECK \(p. 554\)](#)
- [IOT\\_ROLE\\_ALIAS\\_ALLOWS\\_ACCESS\\_TO\\_UNUSED\\_SERVICES\\_CHECK \(p. 555\)](#)
- [CA\\_CERT\\_APPROACHING\\_EXPIRATION\\_CHECK \(p. 556\)](#)
- [CONFLICTING\\_CLIENT\\_IDS\\_CHECK \(p. 557\)](#)
- [DEVICE\\_CERT\\_APPROACHING\\_EXPIRATION\\_CHECK \(p. 557\)](#)
- [REVOKED\\_DEVICE\\_CERT\\_CHECK \(p. 558\)](#)
- [LOGGING\\_DISABLED\\_CHECK \(p. 559\)](#)

## REVOKED\_CA\_CERT\_CHECK

A CA certificate was revoked, but is still active in AWS IoT.

Severity: **Critical**

## Details

A CA certificate is marked as revoked in the certificate revocation list maintained by the issuing authority, but is still marked as ACTIVE or PENDING\_TRANSFER in AWS IoT.

The following reason codes are returned when this check finds a noncompliant CA certificate:

- CERTIFICATE\_REVOKED\_BY\_ISSUER

## Why it matters

A revoked CA certificate should no longer be used to sign device certificates. It might have been revoked because it was compromised. Newly added devices with certificates signed using this CA certificate might pose a security threat.

## How to fix it

1. Use [UpdateCACertificate](#) to mark the CA certificate as INACTIVE in AWS IoT. You can also use mitigation actions to:
  - Apply the UPDATE\_CA\_CERTIFICATE mitigation action on your audit findings to make this change.
  - Apply the PUBLISH\_FINDINGS\_TO\_SNS mitigation action to implement a custom response in response to the Amazon SNS message.For more information, see [Mitigation Actions \(p. 592\)](#).
2. Review the device certificate registration activity for the time after the CA certificate was revoked and consider revoking any device certificates that might have been issued with it during this time. You can use [ListCertificatesByCA](#) to list the device certificates signed by the CA certificate and [UpdateCertificate](#) to revoke a device certificate.

## DEVICE\_CERTIFICATE\_SHARED\_CHECK

Multiple, concurrent connections use the same X.509 certificate to authenticate with AWS IoT.

Severity: **Critical**

## Details

When this check is enabled, data collection starts immediately, but results of the check are not available for at least two hours.

When performed as part of an on-demand audit, this check looks at the certificates and client IDs that were used by devices to connect during the 31 days before the start of the audit. For scheduled audits, this check looks at data from the last time the audit was run to the time this instance of the audit started. If you have taken steps to mitigate this condition during the time checked, note when the concurrent connections were made to determine if the problem persists.

The following reason codes are returned when this check finds a noncompliant certificate:

- CERTIFICATE\_SHARED\_BY\_MULTIPLE\_DEVICES

In addition, the findings returned by this check include the ID of the shared certificate, the IDs of the clients using the certificate to connect, and the connect/disconnect times. Most recent results are listed first.

## Why it matters

Each device should have a unique certificate to authenticate with AWS IoT. When multiple devices use the same certificate, this might indicate that a device has been compromised. Its identity might have been cloned to further compromise the system.

## How to fix it

Verify that the device certificate has not been compromised. If it has, follow your security best practices to mitigate the situation.

If you use the same certificate on multiple devices, you might want to:

1. Provision new, unique certificates and attach them to each device.
2. Verify that the new certificates are valid and the devices can use them to connect.
3. Use [UpdateCertificate](#) to mark the old certificate as REVOKED in AWS IoT. You can also use mitigation actions to do the following:
  - Apply the UPDATE\_DEVICE\_CERTIFICATE mitigation action on your audit findings to make this change.
  - Apply the ADD\_THINGS\_TO\_THING\_GROUP mitigation action to add the device to a group where you can take action on it.
  - Apply the PUBLISH\_FINDINGS\_TO\_SNS mitigation action if you want to implement a custom response in response to the Amazon SNS message.

For more information, see [Mitigation Actions \(p. 592\)](#).

4. Detach the old certificate from each of the devices.

## DEVICE\_CERTIFICATE\_KEY\_QUALITY\_CHECK

AWS IoT customers often rely on TLS mutual authentication using X.509 certificates for authenticating to AWS IoT message broker. These certificates and their certificate authority certificates must be registered in their AWS IoT account before they are used. AWS IoT performs basic sanity checks on these certificates when they are registered. These checks include:

- They must be in a valid format.
- They must be signed by a registered certificate authority.
- They must still be within their validity period (in other words, they haven't expired).
- Their cryptographic key sizes must meet a minimum required size (for RSA keys, they must be 2048 bits or larger).

This audit check provides the following additional tests of the quality of your cryptographic key:

- CVE-2008-0166 – Check whether the key was generated using OpenSSL 0.9.8c-1 up to versions before 0.9.8g-9 on a Debian-based operating system. Those versions of OpenSSL use a random number generator that generates predictable numbers, making it easier for remote attackers to conduct brute force guessing attacks against cryptographic keys.
- CVE-2017-15361 – Check whether the key was generated by the Infineon RSA library 1.02.013 in Infineon Trusted Platform Module (TPM) firmware, such as versions before 000000000000422 – 4.34, before 000000000000062b – 6.43, and before 0000000000008521 – 133.33. That library mishandles RSA key generation, making it easier for attackers to defeat some cryptographic protection mechanisms through targeted attacks. Examples of affected technologies include BitLocker with TPM 1.2, YubiKey 4 (before 4.3.5) PGP key generation, and the Cached User Data encryption feature in Chrome OS.

AWS IoT Device Defender reports certificates as noncompliant if they fail these tests.

Severity: **Critical**

## Details

This check applies to device certificates that are ACTIVE or PENDING\_TRANSFER.

The following reason codes are returned when this check finds a noncompliant certificate:

- CERTIFICATE\_KEY\_VULNERABILITY\_CVE-2017-15361
- CERTIFICATE\_KEY\_VULNERABILITY\_CVE-2008-0166

## Why it matters

When a device uses a vulnerable certificate, attackers can more easily compromise that device.

## How to fix it

Update your device certificates to replace those with known vulnerabilities.

If you are using the same certificate on multiple devices, you might want to:

1. Provision new, unique certificates and attach them to each device.
2. Verify that the new certificates are valid and the devices can use them to connect.
3. Use [UpdateCertificate](#) to mark the old certificate as REVOKED in AWS IoT. You can also use mitigation actions to:
  - Apply the UPDATE\_DEVICE\_CERTIFICATE mitigation action on your audit findings to make this change.
  - Apply the ADD\_THINGS\_TO\_THING\_GROUP mitigation action to add the device to a group where you can take action on it.
  - Apply the PUBLISH\_FINDINGS\_TO\_SNS mitigation action if you want to implement a custom response in response to the Amazon SNS message.

For more information, see [Mitigation Actions \(p. 592\)](#).

4. Detach the old certificate from each of the devices.

## CA\_CERTIFICATE\_KEY\_QUALITY\_CHECK

AWS IoT customers often rely on TLS mutual authentication using X.509 certificates for authenticating to AWS IoT message broker. These certificates and their certificate authority certificates must be registered in their AWS IoT account before they are used. AWS IoT performs basic sanity checks on these certificates when they are registered, including:

- The certificates are in a valid format.
- The certificates are within their validity period (in other words, not expired).
- Their cryptographic key sizes meet a minimum required size (for RSA keys, they must be 2048 bits or larger).

This audit check provides the following additional tests of the quality of your cryptographic key:

- CVE-2008-0166 – Check whether the key was generated using OpenSSL 0.9.8c-1 up to versions before 0.9.8g-9 on a Debian-based operating system. Those versions of OpenSSL use a random number

generator that generates predictable numbers, making it easier for remote attackers to conduct brute force guessing attacks against cryptographic keys.

- CVE-2017-15361 – Check whether the key was generated by the Infineon RSA library 1.02.013 in Infineon Trusted Platform Module (TPM) firmware, such as versions before 0000000000000422 – 4.34, before 000000000000062b – 6.43, and before 0000000000008521 – 133.33. That library mishandles RSA key generation, making it easier for attackers to defeat some cryptographic protection mechanisms through targeted attacks. Examples of affected technologies include BitLocker with TPM 1.2, YubiKey 4 (before 4.3.5) PGP key generation, and the Cached User Data encryption feature in Chrome OS.

AWS IoT Device Defender reports certificates as noncompliant if they fail these tests.

Severity: **Critical**

## Details

This check applies to CA certificates that are ACTIVE or PENDING\_TRANSFER.

The following reason codes are returned when this check finds a noncompliant certificate:

- CERTIFICATE\_KEY\_VULNERABILITY\_CVE-2017-15361
- CERTIFICATE\_KEY\_VULNERABILITY\_CVE-2008-0166

## Why it matters

Newly added devices signed using this CA certificate might pose a security threat.

## How to fix it

1. Use [UpdateCACertificate](#) to mark the CA certificate as INACTIVE in AWS IoT. You can also use mitigation actions to:
  - Apply the `UPDATE_CA_CERTIFICATE` mitigation action on your audit findings to make this change.
  - Apply the `PUBLISH_FINDINGS_TO_SNS` mitigation action if you want to implement a custom response in response to the Amazon SNS message.For more information, see [Mitigation Actions \(p. 592\)](#).
2. Review the device certificate registration activity for the time after the CA certificate was revoked and consider revoking any device certificates that might have been issued with it during this time. (Use [ListCertificatesByCA](#) to list the device certificates signed by the CA certificate and [UpdateCertificate](#) to revoke a device certificate.)

## UNAUTHENTICATED\_COGNITO\_ROLE\_OVERLY\_PERMISSIVE\_CHECK

A policy attached to an unauthenticated Amazon Cognito identity pool role is considered too permissive because it grants permission to perform any of the following AWS IoT actions:

- Manage or modify things.
- Read thing administrative data.
- Manage non-thing related data or resources.

Or, because it grants permission to perform the following AWS IoT actions on a broad set of devices:

- Use MQTT to connect, publish, or subscribe to reserved topics (including shadow or job execution data).

- Use API commands to read or modify shadow or job execution data.

In general, devices that connect using an unauthenticated Amazon Cognito identity pool role should have only limited permission to publish and subscribe to thing-specific MQTT topics or use the API commands to read and modify thing-specific data related to shadow or job execution data.

Severity: **Critical**

## Details

For this check, AWS IoT Device Defender audits all Amazon Cognito identity pools that have been used to connect to the AWS IoT message broker during the 31 days before the audit execution. All Amazon Cognito identity pools from which either an authenticated or unauthenticated Amazon Cognito identity connected are included in the audit.

The following reason codes are returned when this check finds a noncompliant unauthenticated Amazon Cognito identity pool role:

- `ALLOWS_ACCESS_TO_IOT_ADMIN_ACTIONS`
- `ALLOWS_BROAD_ACCESS_TO_IOT_DATA_PLANE_ACTIONS`

## Why it matters

Because unauthenticated identities are never authenticated by the user, they pose a much greater risk than authenticated Amazon Cognito identities. If an unauthenticated identity is compromised, it can use administrative actions to modify account settings, delete resources, or gain access to sensitive data. Or, with broad access to device settings, it can access or modify shadows and jobs for all devices in your account. A guest user might use the permissions to compromise your entire fleet or launch a DDOS attack with messages.

## How to fix it

A policy attached to an unauthenticated Amazon Cognito identity pool role should grant only those permissions required for a device to do its job. We recommend the following steps:

1. Create a new compliant role.
2. Create a Amazon Cognito identity pool and attach the compliant role to it.
3. Verify that your identities can access AWS IoT using the new pool.
4. After verification is complete, attach the compliant role to the Amazon Cognito identity pool that was flagged as noncompliant.

You can also use mitigation actions to:

- Apply the `PUBLISH_FINDINGS_TO_SNS` mitigation action to implement a custom response in response to the Amazon SNS message.

For more information, see [Mitigation Actions \(p. 592\)](#).

## Manage or modify things

The following AWS IoT API actions are used to manage or modify things. Permission to perform these actions should not be granted to devices that connect through an unauthenticated Amazon Cognito identity pool.

- `AddThingToThingGroup`

- `AttachThingPrincipal`
- `CreateThing`
- `DeleteThing`
- `DetachThingPrincipal`
- `ListThings`
- `ListThingsInThingGroup`
- `RegisterThing`
- `RemoveThingFromThingGroup`
- `UpdateThing`
- `UpdateThingGroupsForThing`

Any role that grants permission to perform these actions on even a single resource is considered noncompliant.

## Read thing administrative data

The following AWS IoT API actions are used to read or modify thing data. Devices that connect through an unauthenticated Amazon Cognito identity pool should not be given permission to perform these actions.

- `DescribeThing`
- `ListJobExecutionsForThing`
- `ListThingGroupsForThing`
- `ListThingPrincipals`

### Example

- noncompliant:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeThing",
                "iot>ListJobExecutionsForThing",
                "iot>ListThingGroupsForThing",
                "iot>ListThingPrincipals"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:/thing/MyThing"
            ]
        }
    ]
}
```

This allows the device to perform the specified actions even though it is granted for one thing only.

## Manage non-things

Devices that connect through an unauthenticated Amazon Cognito identity pool should not be given permission to perform AWS IoT API actions other than those discussed in these sections. You can manage

your account with an application that connects through an unauthenticated Amazon Cognito identity pool by creating a separate identity pool not used by devices.

## Subscribe/publish to MQTT topics

MQTT messages are sent through the AWS IoT message broker and are used by devices to perform many actions, including accessing and modifying shadow state and job execution state. A policy that grants permission to a device to connect, publish, or subscribe to MQTT messages should restrict these actions to specific resources as follows:

### Connect

- noncompliant:

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

The wildcard \* allows any device to connect to AWS IoT.

```
arn:aws:iot:region:account-id:client/${iot:ClientId}
```

Unless `iot:Connection.Thing.IsAttached` is set to true in the condition keys, this is equivalent to the wildcard \* in the previous example.

- compliant:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "iot:Connect" ],
      "Resource": [
        "arn:aws:iot:region:account-id:client/${iot:Connection.Thing.ThingName}"
      ],
      "Condition": {
        "Bool": { "iot:Connection.Thing.IsAttached": "true" }
      }
    }
  ]
}
```

The resource specification contains a variable that matches the device name used to connect. The condition statement further restricts the permission by checking that the certificate used by the MQTT client matches that attached to the thing with the name used.

### Publish

- noncompliant:

```
arn:aws:iot:region:account-id:topic/$aws/things/*/shadow/update
```

This allows the device to update the shadow of any device (\* = all devices).

```
arn:aws:iot:region:account-id:topic/$aws/things/*
```

This allows the device to read, update, or delete the shadow of any device.

- compliant:

```
{
```

```

    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [ "iot:Publish" ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/*"
            ],
        }
    ]
}

```

The resource specification contains a wildcard, but it only matches any shadow-related topic for the device whose thing name is used to connect.

### Subscribe

- noncompliant:

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

This allows the device to subscribe to reserved shadow or job topics for all devices.

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

The same as the previous example, but using the # wildcard.

```
arn:aws:iot:region:account-id:topic/$aws/things/#/shadow/update
```

This allows the device to see shadow updates on any device (+ = all devices).

- compliant:

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [ "iot:Subscribe" ],
            "Resource": [
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/*"
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/jobs/*"
            ],
        }
    ]
}

```

The resource specifications contain wildcards, but they only match any shadow-related topic and any job-related topic for the device whose thing name is used to connect.

### Receive

- compliant:

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

This is allowed because the device can receive messages only from topics on which it has permission to subscribe.

## Read/modify shadow or job data

A policy that grants permission to a device to perform an API action to access or modify device shadows or job execution data should restrict these actions to specific resources. The following are the API actions:

- `DeleteThingShadow`
- `GetThingShadow`
- `UpdateThingShadow`
- `DescribeJobExecution`
- `GetPendingJobExecutions`
- `StartNextPendingJobExecution`
- `UpdateJobExecution`

### Example

- noncompliant:

```
arn:aws:iot:region:account-id:thing/*
```

This allows the device to perform the specified action on any thing.

- compliant:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:DeleteThingShadow",
                "iot:GetThingShadow",
                "iot:UpdateThingShadow",
                "iot:DescribeJobExecution",
                "iot:GetPendingJobExecutions",
                "iot:StartNextPendingJobExecution",
                "iot:UpdateJobExecution"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:thing/MyThing1",
                "arn:aws:iot:region:account-id:thing/MyThing2"
            ]
        }
    ]
}
```

This allows the device to perform the specified actions on two things only.

## AUTHENTICATED\_COGNITO\_ROLE\_OVERLY\_PERMISSIVE\_CHECK

A policy attached to an authenticated Amazon Cognito identity pool role is considered too permissive because it grants permission to perform the following AWS IoT actions:

- Manage or modify things.
- Manage non-thing related data or resources.

Or, because it grants permission to perform the following AWS IoT actions on a broad set of devices:

- Read thing administrative data.
- Use MQTT to connect/publish/subscribe to reserved topics (including shadow or job execution data).
- Use API commands to read or modify shadow or job execution data.

In general, devices that connect using an authenticated Amazon Cognito identity pool role should have only limited permission to read thing-specific administrative data, publish and subscribe to thing-specific MQTT topics, or use the API commands to read and modify thing-specific data related to shadow or job execution data.

Severity: **Critical**

## Details

For this check, AWS IoT Device Defender audits all Amazon Cognito identity pools that have been used to connect to the AWS IoT message broker during the 31 days before the audit execution. All Amazon Cognito identity pools from which either an authenticated or unauthenticated Amazon Cognito identity connected are included in the audit.

The following reason codes are returned when this check finds a noncompliant authenticated Amazon Cognito identity pool role:

- ALLOWS\_BROAD\_ACCESS\_TO\_IOT\_THING\_ADMIN\_READ\_ACTIONS
- ALLOWS\_ACCESS\_TO\_IOT\_NON\_THING\_ADMIN\_ACTIONS
- ALLOWS\_ACCESS\_TO\_IOT\_THING\_ADMIN\_WRITE\_ACTIONS

## Why it matters

If an authenticated identity is compromised, it can use administrative actions to modify account settings, delete resources, or gain access to sensitive data.

## How to fix it

A policy attached to an authenticated Amazon Cognito identity pool role should grant only those permissions required for a device to do its job. We recommend the following steps:

1. Create a new compliant role.
2. Create a Amazon Cognito identity pool and attach the compliant role to it.
3. Verify that your identities can access AWS IoT using the new pool.
4. After verification is complete, attach the role to the Amazon Cognito identity pool that was flagged as noncompliant.

You can also use mitigation actions to:

- Apply the PUBLISH\_FINDINGS\_TO\_SNS mitigation action to implement a custom response in response to the Amazon SNS message.

For more information, see [Mitigation Actions \(p. 592\)](#).

## Manage or modify things

The following AWS IoT API actions are used to manage or modify things so permission to perform these should not be granted to devices connecting through an authenticated Amazon Cognito identity pool:

- AddThingToThingGroup
- AttachThingPrincipal
- CreateThing
- DeleteThing
- DetachThingPrincipal
- ListThings
- ListThingsInThingGroup
- RegisterThing
- RemoveThingFromThingGroup
- UpdateThing
- UpdateThingGroupsForThing

Any role that grants permission to perform these actions on even a single resource is considered noncompliant.

## Manage non-things

Devices that connect through an authenticated Amazon Cognito identity pool should not be given permission to perform AWS IoT API actions other than those discussed in these sections. To manage your account with an application that connects through an authenticated Amazon Cognito identity pool, create a separate identity pool not used by devices.

## Read thing administrative data

The following AWS IoT API actions are used to read thing data, so devices that connect through an authenticated Amazon Cognito identity pool should be given permission to perform these on a limited set of things only:

- DescribeThing
- ListJobExecutionsForThing
- ListThingGroupsForThing
- ListThingPrincipals

- noncompliant:

```
arn:aws:iot:region:account-id:thing/*
```

This allows the device to perform the specified action on any thing.

- compliant:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iot:DescribeThing",  
                "iot>ListJobExecutionsForThing",  
                "iot>ListThingGroupsForThing",  
                "iot>ListThingPrincipals"  
            ],  
            "Resource": [  
                "arn:aws:iot:region:account-id:/thing/MyThing"  
            ]  
        }  
    ]  
}
```

```

        ]
    }
}
```

This allows the device to perform the specified actions on only one thing.

- compliant:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeThing",
                "iot>ListJobExecutionsForThing",
                "iot>ListThingGroupsForThing",
                "iot>ListThingPrincipals"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:thing/MyThing*"
            ]
        }
    ]
}
```

This is compliant because, although the resource is specified with a wildcard (\*), it is preceded by a specific string, and that limits the set of things accessed to those with names that have the given prefix.

- noncompliant:

```
arn:aws:iot:region:account-id:thing/*
```

This allows the device to perform the specified action on any thing.

- compliant:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeThing",
                "iot>ListJobExecutionsForThing",
                "iot>ListThingGroupsForThing",
                "iot>ListThingPrincipals"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:thing/MyThing"
            ]
        }
    ]
}
```

This allows the device to perform the specified actions on only one thing.

- compliant:

```
{
```

```

    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeThing",
                "iot>ListJobExecutionsForThing",
                "iot>ListThingGroupsForThing",
                "iot>ListThingPrincipals"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:thing/MyThing*"
            ]
        }
    ]
}

```

This is compliant because, although the resource is specified with a wildcard (\*), it is preceded by a specific string, and that limits the set of things accessed to those with names that have the given prefix.

## Subscribe/publish to MQTT topics

MQTT messages are sent through the AWS IoT message broker and are used by devices to perform many different actions, including accessing and modifying shadow state and job execution state. A policy that grants permission to a device to connect, publish, or subscribe to MQTT messages should restrict these actions to specific resources as follows:

### Connect

- noncompliant:

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

The wildcard \* allows any device to connect to AWS IoT.

```
arn:aws:iot:region:account-id:client/${iot:ClientId}
```

Unless `iot:Connection.Thing.IsAttached` is set to true in the condition keys, this is equivalent to the wildcard \* in the previous example.

- compliant:

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [ "iot:Connect" ],
            "Resource": [
                "arn:aws:iot:region:account-id:client/${iot:Connection.Thing.ThingName}"
            ],
            "Condition": {
                "Bool": { "iot:Connection.Thing.IsAttached": "true" }
            }
        }
    ]
}

```

The resource specification contains a variable that matches the device name used to connect, and the condition statement further restricts the permission by checking that the certificate used by the MQTT client matches that attached to the thing with the name used.

### Publish

- noncompliant:

```
arn:aws:iot:region:account-id:topic/$aws/things/*/shadow/update
```

This allows the device to update the shadow of any device (\* = all devices).

```
arn:aws:iot:region:account-id:topic/$aws/things/*
```

This allows the device to read/update/delete the shadow of any device.

- compliant:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [ "iot:Publish" ],  
            "Resource": [  
                "arn:aws:iot:region:account-id:topic/$aws/things/  
                ${iot:Connection.Thing.ThingName}/shadow/*"  
            ]  
        }  
    ]  
}
```

The resource specification contains a wildcard, but it only matches any shadow-related topic for the device whose thing name is used to connect.

### Subscribe

- noncompliant:

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

This allows the device to subscribe to reserved shadow or job topics for all devices.

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/#
```

The same as the previous example, but using the # wildcard.

```
arn:aws:iot:region:account-id:topic/$aws/things/+shadow/update
```

This allows the device to see shadow updates on any device (+ = all devices).

- compliant:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [ "iot:Subscribe" ],  
            "TopicFilter": "$aws/things/+shadow/update"
```

```

    "Resource": [
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
        ${iot:Connection.Thing.ThingName}/shadow/*"
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
        ${iot:Connection.Thing.ThingName}/jobs/*"
    ],
}
]
}

```

The resource specifications contain wildcards, but they only match any shadow-related topic and any job-related topic for the device whose thing name is used to connect.

#### Receive

- compliant:

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

This is compliant because the device can receive messages only from topics on which it has permission to subscribe.

#### Read or modify shadow or job data

A policy that grants permission to a device to perform an API action to access or modify device shadows or job execution data should restrict these actions to specific resources. The following are the API actions:

- `DeleteThingShadow`
- `GetThingShadow`
- `UpdateThingShadow`
- `DescribeJobExecution`
- `GetPendingJobExecutions`
- `StartNextPendingJobExecution`
- `UpdateJobExecution`

#### Examples

- noncompliant:

```
arn:aws:iot:region:account-id:thing/*
```

This allows the device to perform the specified action on any thing.

- compliant:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:DeleteThingShadow",
                "iot:GetThingShadow",
                "iot:UpdateThingShadow",
                "iot:DescribeJobExecution",
                "iot:GetPendingJobExecutions",

```

```
        "iot:StartNextPendingJobExecution",
        "iot:UpdateJobExecution"
    ],
    "Resource": [
        "arn:aws:iot:region:account-id:thing/MyThing1",
        "arn:aws:iot:region:account-id:thing/MyThing2"
    ]
}
}
```

This allows the device to perform the specified actions on only two things.

## IOT\_POLICY\_OVERLY\_PERMISSIVE\_CHECK

An AWS IoT policy gives permissions that are too broad or unrestricted. It grants permission to send or receive MQTT messages for a broad set of devices, or grants permission to access or modify shadow and job execution data for a broad set of devices.

In general, a policy for a device should grant access to resources associated with just that device and no or very few other devices. With some exceptions, using a wildcard (for example, "") to specify resources in such a policy is considered too broad or unrestricted.

Severity: **Critical**

### Details

The following reason code is returned when this check finds a noncompliant AWS IoT policy:

- ALLOWS\_BROAD\_ACCESS\_TO\_IOT\_DATA\_PLANE\_ACTIONS

### Why it matters

A certificate, Amazon Cognito identity, or thing group with an overly permissive policy can, if compromised, impact the security of your entire account. An attacker could use such broad access to read or modify shadows, jobs, or job executions for all your devices. Or an attacker could use a compromised certificate to connect malicious devices or launch a DDOS attack on your network.

### How to fix it

Follow these steps to fix any noncompliant policies attached to things, thing groups, or other entities:

1. Use [CreatePolicyVersion](#) to create a new, compliant version of the policy. Set the `setAsDefault` flag to true. (This makes this new version operative for all entities that use the policy.)
2. Use [ListTargetsForPolicy](#) to get a list of targets (certificates, thing groups) that the policy is attached to and determine which devices are included in the groups or which use the certificates to connect.
3. Verify that all associated devices are able to connect to AWS IoT. If a device is unable to connect, use [SetPolicyVersion](#) to roll back the default policy to the previous version, revise the policy, and try again.

You can use mitigation actions to:

- Apply the `REPLACE_DEFAULT_POLICY_VERSION` mitigation action on your audit findings to make this change.
- Apply the `PUBLISH_FINDINGS_TO_SNS` mitigation action if you want to implement a custom response in response to the Amazon SNS message.

For more information, see [Mitigation Actions \(p. 592\)](#).

Use [AWS IoT policy variables](#) to dynamically reference AWS IoT resources in your policies.

## MQTT permissions

MQTT messages are sent through the AWS IoT message broker and are used by devices to perform many actions, including accessing and modifying shadow state and job execution state. A policy that grants permission to a device to connect, publish, or subscribe to MQTT messages should restrict these actions to specific resources as follows:

### Connect

- noncompliant:

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

The wildcard \* allows any device to connect to AWS IoT.

```
arn:aws:iot:region:account-id:client/${iot:ClientId}
```

Unless iot:Connection.Thing.IsAttached is set to true in the condition keys, this is equivalent to the wildcard \* as in the previous example.

- compliant:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [ "iot:Connect" ],
            "Resource": [
                "arn:aws:iot:region:account-id:client/${iot:Connection.Thing.ThingName}"
            ],
            "Condition": {
                "Bool": { "iot:Connection.Thing.IsAttached": "true" }
            }
        }
    ]
}
```

The resource specification contains a variable that matches the device name used to connect. The condition statement further restricts the permission by checking that the certificate used by the MQTT client matches that attached to the thing with the name used.

### Publish

- noncompliant:

```
arn:aws:iot:region:account-id:topic/$aws/things/*/shadow/update
```

This allows the device to update the shadow of any device (\* = all devices).

```
arn:aws:iot:region:account-id:topic/$aws/things/*
```

This allows the device to read, update, or delete the shadow of any device.

- compliant:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "iot:Publish" ],
      "Resource": [
        "arn:aws:iot:region:account-id:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/*"
      ],
    }
  ]
}
```

The resource specification contains a wildcard, but it only matches any shadow-related topic for the device whose thing name is used to connect.

### Subscribe

- noncompliant:

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

This allows the device to subscribe to reserved shadow or job topics for all devices.

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

The same as the previous example, but using the # wildcard.

```
arn:aws:iot:region:account-id:topic/$aws/things/#/shadow/update
```

This allows the device to see shadow updates on any device (+ = all devices).

- compliant:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "iot:Subscribe" ],
      "Resource": [
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/*"
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/jobs/*"
      ],
    }
  ]
}
```

The resource specifications contain wildcards, but they only match any shadow-related topic and any job-related topic for the device whose thing name is used to connect.

### Receive

- compliant:

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

This is compliant because the device can only receive messages from topics on which it has permission to subscribe.

## Shadow and job permissions

A policy that grants permission to a device to perform an API action to access or modify device shadows or job execution data should restrict these actions to specific resources. The following are the API actions:

- DeleteThingShadow
- GetThingShadow
- UpdateThingShadow
- DescribeJobExecution
- GetPendingJobExecutions
- StartNextPendingJobExecution
- UpdateJobExecution

### Examples

- noncompliant:

```
arn:aws:iot:region:account-id:thing/*
```

This allows the device to perform the specified action on any thing.

- compliant:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:DeleteThingShadow",
                "iot:GetThingShadow",
                "iot:UpdateThingShadow",
                "iot:DescribeJobExecution",
                "iot:GetPendingJobExecutions",
                "iot:StartNextPendingJobExecution",
                "iot:UpdateJobExecution"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:thing/MyThing1",
                "arn:aws:iot:region:account-id:thing/MyThing2"
            ]
        }
    ]
}
```

This allows the device to perform the specified actions on only two things.

## IOT\_ROLE\_ALIAS\_OVERLY\_PERMISSIVE\_CHECK

AWS IoT role alias provides a mechanism for connected devices to authenticate to AWS IoT using X.509 certificates and then obtain short-lived AWS credentials from an IAM role that is associated with an AWS

IoT role alias. The permissions for these credentials must be scoped down using access policies with authentication context variables. If your policies are not configured correctly, you could leave yourself exposed to an escalation of privilege attack. This audit check ensures that the temporary credentials provided by AWS IoT role aliases are not overly permissive.

This check is triggered if one of the following conditions are found:

- The policy provides administrative permissions to any services used in the past year by this role alias (for example, "iot:\"", "dynamodb:\"", "iam:\"", and so on).
- The policy provides broad access to thing metadata actions, access to restricted AWS IoT actions, or broad access to AWS IoT data plane actions.
- The policy provides access to security auditing services such as "iam", "cloudtrail", "guardduty", "inspector", or "trustedadvisor".

Severity: **Critical**

## Details

The following reason codes are returned when this check finds a noncompliant IoT policy:

- ALLOWS\_BROAD\_ACCESS\_TO\_USED\_SERVICES
- ALLOWS\_ACCESS\_TO\_SECURITY\_AUDITING\_SERVICES
- ALLOWS\_BROAD\_ACCESS\_TO\_IOT\_THING\_ADMIN\_READ\_ACTIONS
- ALLOWS\_ACCESS\_TO\_IOT\_NON\_THING\_ADMIN\_ACTIONS
- ALLOWS\_ACCESS\_TO\_IOT\_THING\_ADMIN\_WRITE\_ACTIONS
- ALLOWS\_BROAD\_ACCESS\_TO\_IOT\_DATA\_PLANE\_ACTIONS

## Why it matters

By limiting permissions to those that are required for a device to perform its normal operations, you reduce the risks to your account if a device is compromised.

## How to fix it

Follow these steps to fix any noncompliant policies attached to things, thing groups, or other entities:

1. Follow the steps in [Authorizing Direct Calls to AWS Services \(p. 167\)](#) to apply a more restrictive policy to your role alias.

You can use mitigation actions to:

- Apply the PUBLISH\_FINDINGS\_TO\_SNS mitigation action if you want to implement a custom action in response to the Amazon SNS message.

For more information, see [Mitigation Actions \(p. 592\)](#).

## IOT\_ROLE\_ALIAS\_ALLows\_ACCESS\_TO\_UNUSED\_SERVICES\_CHECK

AWS IoT role alias provides a mechanism for connected devices to authenticate to AWS IoT using X.509 certificates and then obtain short-lived AWS credentials from an IAM role that is associated with an AWS IoT role alias. The permissions for these credentials must be scoped down using access policies with authentication context variables. If your policies are not configured correctly, you could leave yourself exposed to an escalation of privilege attack. This audit check ensures that the temporary credentials provided by AWS IoT role aliases are not overly permissive.

This check is triggered if the role alias has access to services that haven't been used for the AWS IoT device in the last year. For example, the audit reports if you have an IAM role linked to the role alias that has only used AWS IoT in the past year but the policy attached to the role also grants permission to "iam:getRole" and "dynamodb:PutItem".

Severity: **Medium**

### Details

The following reason codes are returned when this check finds a noncompliant AWS IoT policy:

- `ALLOWS_ACCESS_TO_UNUSED_SERVICES`

### Why it matters

By limiting permissions to those services that are required for a device to perform its normal operations, you reduce the risks to your account if a device is compromised.

### How to fix it

Follow these steps to fix any noncompliant policies attached to things, thing groups, or other entities:

1. Follow the steps in [Authorizing Direct Calls to AWS Services \(p. 167\)](#) to apply a more restrictive policy to your role alias.

You can use mitigation actions to:

- Apply the `PUBLISH_FINDINGS_TO_SNS` mitigation action if you want to implement a custom action in response to the Amazon SNS message.

For more information, see [Mitigation Actions \(p. 592\)](#).

## CA\_CERT\_APPROACHING\_EXPIRATION\_CHECK

A CA certificate is expiring within 30 days or has expired.

Severity: **Medium**

### Details

This check applies to CA certificates that are ACTIVE or PENDING\_TRANSFER.

The following reason codes are returned when this check finds a noncompliant CA certificate:

- `CERTIFICATE_APPROACHING_EXPIRATION`
- `CERTIFICATE_PAST_EXPIRATION`

### Why it matters

An expired CA certificate should not be used to sign new device certificates.

### How to fix it

Consult your security best practices for how to proceed. You might want to:

1. Register a new CA certificate with AWS IoT.
2. Verify that you are able to sign device certificates using the new CA certificate.

3. Use [UpdateCACertificate](#) to mark the old CA certificate as INACTIVE in AWS IoT. You can also use mitigation actions to do the following:
- Apply the `UPDATE_CA_CERTIFICATE` mitigation action on your audit findings to make this change.
  - Apply the `PUBLISH_FINDINGS_TO_SNS` mitigation action if you want to implement a custom response in response to the Amazon SNS message.

For more information, see [Mitigation Actions \(p. 592\)](#).

## CONFLICTING\_CLIENT\_IDS\_CHECK

Multiple devices connect using the same client ID.

Severity: **High**

### Details

Multiple connections were made using the same client ID, causing an already connected device to be disconnected. The MQTT specification allows only one active connection per client ID, so when another device connects using the same client ID, it knocks the previous one off the connection.

When performed as part of an on-demand audit, this check looks at how client IDs were used to connect during the 31 days prior to the start of the audit. For scheduled audits, this check looks at data from the last time the audit was run to the time this instance of the audit started. If you have taken steps to mitigate this condition during the time checked, note when the connections/disconnections were made to determine if the problem persists.

The following reason codes are returned when this check finds noncompliance:

- `DUPLICATE_CLIENT_ID_ACROSS_CONNECTIONS`

The findings returned by this check also include the client ID used to connect, principal IDs, and disconnect times. The most recent results are listed first.

### Why it matters

Devices with conflicting IDs are forced to constantly reconnect, which might result in lost messages or leave a device unable to connect.

This might indicate that a device or a device's credentials have been compromised, and might be part of a DDoS attack. It is also possible that devices are not configured correctly in the account or a device has a bad connection and is forced to reconnect several times per minute.

### How to fix it

Register each device as a unique thing in AWS IoT, and use the thing name as the client ID to connect. Or use a UUID as the client ID when connecting the device over MQTT. You can also use mitigation actions to:

- Apply the `PUBLISH_FINDINGS_TO_SNS` mitigation action if you want to implement a custom response in response to the Amazon SNS message.

For more information, see [Mitigation Actions \(p. 592\)](#).

## DEVICE\_CERT\_APPROACHING\_EXPIRATION\_CHECK

A device certificate is expiring within 30 days or has expired.

Severity: **Medium**

## Details

This check applies to device certificates that are ACTIVE or PENDING\_TRANSFER.

The following reason codes are returned when this check finds a noncompliant device certificate:

- CERTIFICATE\_APPROACHING\_EXPIRATION
- CERTIFICATE\_PAST\_EXPIRATION

## Why it matters

A device certificate should not be used after it expires.

## How to fix it

Consult your security best practices for how to proceed. You might want to:

1. Provision a new certificate and attach it to the device.
2. Verify that the new certificate is valid and the device is able to use it to connect.
3. Use [UpdateCertificate](#) to mark the old certificate as INACTIVE in AWS IoT. You can also use mitigation actions to:
  - Apply the UPDATE\_DEVICE\_CERTIFICATE mitigation action on your audit findings to make this change.
  - Apply the ADD\_THINGS\_TO\_THING\_GROUP mitigation action to add the device to a group where you can take action on it.
  - Apply the PUBLISH\_FINDINGS\_TO\_SNS mitigation action if you want to implement a custom response in response to the Amazon SNS message.

For more information, see [Mitigation Actions \(p. 592\)](#).

4. Detach the old certificate from the device. (See [DetachThingPrincipal](#).)

## REVOKED\_DEVICE\_CERT\_CHECK

A revoked device certificate is still active.

Severity: **Medium**

## Details

A device certificate is in its CA's [certificate revocation list](#), but it is still active in AWS IoT.

This check applies to device certificates that are ACTIVE or PENDING\_TRANSFER.

The following reason codes are returned when this check finds noncompliance:

- CERTIFICATE\_REVOKED\_BY\_ISSUER

## Why it matters

A device certificate is usually revoked because it has been compromised. It is possible that it has not yet been revoked in AWS IoT due to an error or oversight.

## How to fix it

Verify that the device certificate has not been compromised. If it has, follow your security best practices to mitigate the situation. You might want to:

1. Provision a new certificate for the device.
2. Verify that the new certificate is valid and the device is able to use it to connect.
3. Use [UpdateCertificate](#) to mark the old certificate as REVOKED in AWS IoT. You can also use mitigation actions to:
  - Apply the `UPDATE_DEVICE_CERTIFICATE` mitigation action on your audit findings to make this change.
  - Apply the `ADD_THINGS_TO_THING_GROUP` mitigation action to add the device to a group where you can take action on it.
  - Apply the `PUBLISH_FINDINGS_TO_SNS` mitigation action if you want to implement a custom response in response to the Amazon SNS message.

For more information, see [Mitigation Actions \(p. 592\)](#).

4. Detach the old certificate from the device. (See [DetachThingPrincipal](#).)

## LOGGING\_DISABLED\_CHECK

AWS IoT logs are not enabled in Amazon CloudWatch.

Severity: **Low**

### Details

The following reason codes are returned when this check finds noncompliance:

- LOGGING\_DISABLED

### Why it matters

AWS IoT logs in CloudWatch provide visibility into behaviors in AWS IoT, including authentication failures and unexpected connects and disconnects that might indicate that a device has been compromised.

## How to fix it

Enable AWS IoT logs in CloudWatch. See [Monitoring Tools](#). You can also use mitigation actions to:

- Apply the `ENABLE_IOT_LOGGING` mitigation action on your audit findings to make this change.
- Apply the `PUBLISH_FINDINGS_TO_SNS` mitigation action if you want to implement a custom response in response to the Amazon SNS message.

For more information, see [Mitigation Actions \(p. 592\)](#).

## How to Perform Audits

1. Configure audit settings for your account. Use [UpdateAccountAuditConfiguration \(p. 566\)](#) to enable those checks you want to be available for audits, set up optional notifications, and configure permissions.

For some checks, AWS IoT starts collecting data as soon as the check is enabled.

2. Create one or more audit schedules. Use [CreateScheduledAudit \(p. 570\)](#) to specify the checks you want to perform during an audit and how often these audits should be run.

Or, you can run an on-demand audit when necessary. Use [StartOnDemandAuditTask \(p. 579\)](#) to specify the checks you want to perform and start an audit running right away. (Results might not be ready for some time if you have recently enabled a check that is included in the on-demand audit.)

3. You can use the [AWS IoT console](#) to view the results of your audits.

Or, you can see the results of your audits with [ListAuditFindings \(p. 587\)](#). With this command, you can filter the results by the type of check, a specific resource, or the time of the audit. You can use this information to mitigate any problems found.

4. You can apply mitigation actions that you defined in your AWS account to any noncompliant findings. For more information, see [Apply Mitigation Actions \(p. 598\)](#).

## Notifications

When an audit is completed, an SNS notification can be sent with a summary of the results of each audit check that was performed, including details about the number of noncompliant resources that were found. Use the `auditNotificationTargetConfigurations` field in the input to the [UpdateAccountAuditConfiguration \(p. 566\)](#) command. The SNS notification has the following payload:

payload example

```
{  
    "accountId": "123456789012",  
    "taskId": "4e2bcd1ccbc2a5dd15292a82ab80c380",  
    "taskStatus": "FAILED|CANCELED|COMPLETED",  
    "taskType": "ON_DEMAND_AUDIT_TASK|SCHEDULED_AUDIT_TASK",  
    "scheduledAuditName": "myWeeklyAudit",  
    "failedChecksCount": 0,  
    "canceledChecksCount": 0,  
    "nonCompliantChecksCount": 1,  
    "compliantChecksCount": 0,  
    "totalChecksCount": 1,  
    "taskStartTime": 1524740766191,  
    "auditDetails": [  
        {  
            "checkName": "DEVICE_CERT_APPROACHING_EXPIRATION_CHECK |  
                REVOKED_DEVICE_CERT_CHECK |  
                CA_CERT_APPROACHING_EXPIRATION_CHECK |  
                REVOKED_CA_CERT_CHECK |  
                DEVICE_CERTIFICATE_SHARED_CHECK |  
                IOT_POLICY_UNRESTRICTED_CHECK |  
                UNAUTHENTICATED_COGNITO_IDENTITY_UNRESTRICTED_ACCESS_CHECK |  
                AUTHENTICATED_COGNITO_IDENTITY_UNRESTRICTED_ACCESS_CHECK |  
                CONFLICTING_CLIENT_IDS_CHECK |  
                LOGGING_DISABLED_CHECK",  
  
            "checkRunStatus": "FAILED |  
                CANCELED |  
                COMPLETED_COMPLIANT |  
                COMPLETED_NON_COMPLIANT",  
  
            "nonCompliantResourcesCount": 1,  
            "totalResourcesCount": 1,  
  
            "message": "optional message if an error occurred",  
            "errorCode": "INSUFFICIENT_PERMISSIONS|AUDIT_CHECK_DISABLED"  
        }  
    ]  
}
```

}

payload JSON schema

```
{  
    "$schema": "http://json-schema.org/draft-07/schema#",  
    "$id": "arn:aws:iot::::schema:auditnotification/1.0",  
    "type": "object",  
    "properties": {  
        "accountId": {  
            "type": "string"  
        },  
        "taskId": {  
            "type": "string"  
        },  
        "taskStatus": {  
            "type": "string",  
            "enum": [  
                "FAILED",  
                "CANCELED",  
                "COMPLETED"  
            ]  
        },  
        "taskType": {  
            "type": "string",  
            "enum": [  
                "SCHEDULED_AUDIT_TASK",  
                "ON_DEMAND_AUDIT_TASK"  
            ]  
        },  
        "scheduledAuditName": {  
            "type": "string"  
        },  
        "failedChecksCount": {  
            "type": "integer"  
        },  
        "canceledChecksCount": {  
            "type": "integer"  
        },  
        "nonCompliantChecksCount": {  
            "type": "integer"  
        },  
        "compliantChecksCount": {  
            "type": "integer"  
        },  
        "totalChecksCount": {  
            "type": "integer"  
        },  
        "taskStartTime": {  
            "type": "integer"  
        },  
        "auditDetails": {  
            "type": "array",  
            "items": [  
                {  
                    "type": "object",  
                    "properties": {  
                        "checkName": {  
                            "type": "string",  
                            "enum": [  
                                "DEVICE_CERT_APPROACHING_EXPIRATION_CHECK",  
                                "REVOKED_DEVICE_CERT_CHECK",  
                                "CA_CERT_APPROACHING_EXPIRATION_CHECK",  
                                "REVOKED_CA_CERT_CHECK",  
                                "AWS_IOT_CERTIFICATE_EXPIRATION_CHECK",  
                                "REVOKED_AWS_IOT_CERTIFICATE_CHECK"  
                            ]  
                        }  
                    }  
                }  
            ]  
        }  
    }  
}
```

```
        "LOGGING_DISABLED_CHECK"
    ],
},
"checkRunStatus": {
    "type": "string",
    "enum": [
        "FAILED",
        "CANCELED",
        "COMPLETED_COMPLIANT",
        "COMPLETED_NON_COMPLIANT"
    ]
},
"nonCompliantResourcesCount": {
    "type": "integer"
},
"totalResourcesCount": {
    "type": "integer"
},
"message": {
    "type": "string",
},
"errorCode": {
    "type": "string",
    "enum": [
        "INSUFFICIENT_PERMISSIONS",
        "AUDIT_CHECK_DISABLED"
    ]
},
"required": [
    "checkName",
    "checkRunStatus",
    "nonCompliantResourcesCount",
    "totalResourcesCount"
]
}
],
},
"required": [
    "accountId",
    "taskId",
    "taskStatus",
    "taskType",
    "failedChecksCount",
    "canceledChecksCount",
    "nonCompliantChecksCount",
    "compliantChecksCount",
    "totalChecksCount",
    "taskStartTime",
    "auditDetails"
]
}
```

You can also view notifications in the AWS IoT console, along with information about the device, device statistics (for example, last connection time, number of active connections, data transfer rate), and historical alerts for the device.

## Permissions

This section contains information about how to set up the IAM roles and policies required to create, run, and manage AWS IoT Device Defender audits. For more information, see the [AWS Identity and Access Management User Guide](#).

## Give AWS IoT Device Defender permission to collect your data to run an audit

When you call [UpdateAccountAuditConfiguration \(p. 566\)](#), you must specify an IAM role with two policies: a permissions policy and a trust policy. The permissions policy grants AWS IoT Device Defender permission to access your account data, using AWS IoT APIs, when it runs an audit. The trust policy grants AWS IoT Device Defender permission to assume the required role.

### permissions policy

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iot:GetLoggingOptions",  
                "iot:GetV2LoggingOptions",  
                "iot>ListCACertificates",  
                "iot>ListCertificates",  
                "iot:DescribeCACertificate",  
                "iot:DescribeCertificate",  
                "iot>ListPolicies",  
                "iot:GetPolicy",  
                "iot:GetEffectivePolicies",  
                "iot>ListRoleAliases",  
                "iot:DescribeRoleAlias",  
                "cognito-identity:GetIdentityPoolRoles",  
                "iam>ListRolePolicies",  
                "iam>ListAttachedRolePolicies",  
                "iam:GetRole",  
                "iam:GetPolicy",  
                "iam:GetPolicyVersion",  
                "iam:GetRolePolicy",  
                "iam:GenerateServiceLastAccessedDetails",  
                "iam:GetServiceLastAccessedDetails"  
            ],  
            "Resource": [  
                "*"  
            ]  
        }  
    ]  
}
```

### trust policy

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "iot.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

## Give AWS IoT Device Defender permission to publish notifications to an SNS topic

If you use the `auditNotificationTargetConfigurations` parameter in [UpdateAccountAuditConfiguration \(p. 566\)](#), you must specify an IAM role with two policies: a permissions policy and a trust policy. The permissions policy grants permission to AWS IoT Device Defender to publish notifications to your SNS topic. The trust policy grants AWS IoT Device Defender permission to assume the required role.

permissions policy

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "sns:Publish"  
            ],  
            "Resource": [  
                "arn:aws:sns:region:account-id:your-topic-name"  
            ]  
        }  
    ]  
}
```

trust policy

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "iot.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

## Give IAM users or groups permission to run AWS IoT Device Defender audit commands

To allow IAM users or groups to manage, run, or view the results of AWS IoT Device Defender, you must create and assign roles with attached policies that grant permission to run the appropriate commands. The content of each policy depends on which commands you want the user or group to run.

- `UpdateAccountAuditConfiguration`

policy

You must create the IAM role with the attached policy in same account from which this command is run. Cross account access is not allowed. The policy should have `iam:PassRole` permissions (permissions to pass this role).

In the following policy template, `audit-permissions-role-arn` is the role ARN that you pass to AWS IoT Device Defender in the `UpdateAccountAuditConfiguration` request using the

`roleArn` parameter. `audit-notifications-permissions-role-arn` is the role ARN that you pass to AWS IoT Device Defender in the `UpdateAccountAuditConfiguration` request using the `auditNotificationTargetConfigurations` parameter.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateAccountAuditConfiguration"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::account-id:role/audit-permissions-role-arn",
                "arn:aws:iam::account-id:role/audit-notifications-permissions-role-arn"
            ]
        }
    ]
}
```

- `DescribeAccountAuditConfiguration`
- `DeleteAccountAuditConfiguration`
- `StartOnDemandAuditTask`
- `CancelAuditTask`
- `DescribeAuditTask`
- `ListAuditTasks`
- `ListScheduledAudits`
- `ListAuditFindings`

`policy`

All of these commands require \* in the Resource field of the policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeAccountAuditConfiguration",
                "iot:DeleteAccountAuditConfiguration",
                "iot:StartOnDemandAuditTask",
                "iot:CancelAuditTask",
                "iot:DescribeAuditTask",
                "iot>ListAuditTasks",
                "iot>ListScheduledAudits",
                "iot>ListAuditFindings"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

```
        ]
    }
}
```

- CreateScheduledAudit
- UpdateScheduledAudit
- DeleteScheduledAudit
- DescribeScheduledAudit

policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:CreateScheduledAudit",
        "iot:UpdateScheduledAudit",
        "iot:DeleteScheduledAudit",
        "iot:DescribeScheduledAudit"
      ],
      "Resource": [
        "arn:aws:iot:region:account-id:scheduledaudit/scheduled-audit-name"
      ]
    }
  ]
}
```

The format for an AWS IoT Device Defender scheduled audit role ARN is:

```
arn:aws:iot:region:account-id:scheduledaudit/scheduled-audit-name
```

## Audit Commands

### Manage Audit Settings

Use `UpdateAccountAuditConfiguration` to configure audit settings for your account. This command allows you to enable those checks you want to be available for audits, set up optional notifications, and configure permissions.

Check these settings with `DescribeAccountAuditConfiguration`.

Use `DeleteAccountAuditConfiguration` to delete your audit settings. This restores all default values, and effectively disables audits because all checks are disabled by default.

### UpdateAccountAuditConfiguration

Configures or reconfigures the Device Defender audit settings for this account. Settings include how audit notifications are sent and which audit checks are enabled or disabled.

#### Synopsis

```
aws iot update-account-audit-configuration \
[--role-arn <value>] \
[--audit-notification-target-configurations <value>] \
[--audit-check-configurations <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

#### cli-input-json format

```
{
  "roleArn": "string",
  "auditNotificationTargetConfigurations": {
    "string": {
      "targetArn": "string",
      "roleArn": "string",
      "enabled": "boolean"
    }
  },
  "auditCheckConfigurations": {
    "string": {
      "enabled": "boolean"
    }
  }
}
```

#### cli-input-json fields

| Name                                  | Type                              | Description                                                                                                                                                                                                                                                               |
|---------------------------------------|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| roleArn                               | string<br>length- max:2048 min:20 | The ARN of the role that grants permission to AWS IoT to access information about your devices, policies, certificates, and other items when performing an audit.                                                                                                         |
| auditNotificationTargetConfigurations | map                               | Information about the targets to which audit notifications are sent.                                                                                                                                                                                                      |
| targetArn                             | string                            | The ARN of the target (SNS topic) to which audit notifications are sent.                                                                                                                                                                                                  |
| roleArn                               | string<br>length- max:2048 min:20 | The ARN of the role that grants permission to send notifications to the target.                                                                                                                                                                                           |
| enabled                               | boolean                           | True if notifications to the target are enabled.                                                                                                                                                                                                                          |
| auditCheckConfigurations              | map                               | Specifies which audit checks are enabled and disabled for this account. Use <a href="#">DescribeAccountAuditConfiguration</a> to see the list of all checks, including those that are currently enabled.<br><br>Some data collection might start immediately when certain |

| Name    | Type    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         |         | <p>checks are enabled. When a check is disabled, any data collected so far in relation to the check is deleted.</p> <p>You cannot disable a check if it is used by any scheduled audit. You must first delete the check from the scheduled audit or delete the scheduled audit itself.</p> <p>On the first call to <code>UpdateAccountAuditConfiguration</code>, this parameter is required and must specify at least one enabled check.</p> |
| enabled | boolean | True if this audit check is enabled for this account.                                                                                                                                                                                                                                                                                                                                                                                        |

#### Output

None

#### Errors

`InvalidRequestException`

The contents of the request were invalid.

`ThrottlingException`

The rate exceeds the limit.

`InternalFailureException`

An unexpected error has occurred.

## DescribeAccountAuditConfiguration

Gets information about the Device Defender audit settings for this account. Settings include how audit notifications are sent and which audit checks are enabled or disabled.

#### Synopsis

```
aws iot describe-account-audit-configuration \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format

```
{  
}
```

#### Output

```
{
  "roleArn": "string",
  "auditNotificationTargetConfigurations": {
    "string": {
      "targetArn": "string",
      "roleArn": "string",
      "enabled": "boolean"
    }
  },
  "auditCheckConfigurations": {
    "string": {
      "enabled": "boolean"
    }
  }
}
```

### CLI output fields

| Name                                  | Type                              | Description                                                                                                                                                                                                                                                              |
|---------------------------------------|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| roleArn                               | string<br>length- max:2048 min:20 | The ARN of the role that grants permission to AWS IoT to access information about your devices, policies, certificates, and other items when performing an audit.<br><br>On the first call to <code>UpdateAccountAuditConfiguration</code> , this parameter is required. |
| auditNotificationTargetConfigurations | map                               | Information about the targets to which audit notifications are sent for this account.                                                                                                                                                                                    |
| targetArn                             | string                            | The ARN of the target (SNS topic) to which audit notifications are sent.                                                                                                                                                                                                 |
| roleArn                               | string<br>length- max:2048 min:20 | The ARN of the role that grants permission to send notifications to the target.                                                                                                                                                                                          |
| enabled                               | boolean                           | True if notifications to the target are enabled.                                                                                                                                                                                                                         |
| auditCheckConfigurations              | map                               | Which audit checks are enabled and disabled for this account.                                                                                                                                                                                                            |
| enabled                               | boolean                           | True if this audit check is enabled for this account.                                                                                                                                                                                                                    |

### Errors

`ThrottlingException`

The rate exceeds the limit.

`InternalFailureException`

An unexpected error has occurred.

## DeleteAccountAuditConfiguration

Restores the default settings for Device Defender audits for this account. Any configuration data you entered is deleted and all audit checks are reset to disabled.

### Synopsis

```
aws iot delete-account-audit-configuration \
[--delete-scheduled-audits | --no-delete-scheduled-audits] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

**cli-input-json** format

```
{  
  "deleteScheduledAudits": "boolean"  
}
```

### cli-input-json fields

| Name                  | Type    | Description                                |
|-----------------------|---------|--------------------------------------------|
| deleteScheduledAudits | boolean | If true, all scheduled audits are deleted. |

### Output

None

### Errors

**InvalidRequestException**

The contents of the request were invalid.

**ResourceNotFoundException**

The specified resource does not exist.

**ThrottlingException**

The rate exceeds the limit.

**InternalFailureException**

An unexpected error has occurred.

## Schedule Audits

Use `CreateScheduledAudit` to create one or more scheduled audits. This command allows you to specify the checks you want to perform during an audit and how often the audit should be run.

Keep track of your scheduled audits with `ListScheduledAudits` and `DescribeScheduledAudit`.

Change an existing scheduled audit with `UpdateScheduledAudit` or delete it with `DeleteScheduledAudit`.

## CreateScheduledAudit

Creates a scheduled audit that is run at a specified time interval.

## Synopsis

```
aws iot create-scheduled-audit \
--frequency <value> \
[--day-of-month <value>] \
[--day-of-week <value>] \
--target-check-names <value> \
[--tags <value>] \
--scheduled-audit-name <value> \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

## cli-input-json format

```
{
  "frequency": "string",
  "dayOfMonth": "string",
  "dayOfWeek": "string",
  "targetCheckNames": [
    "string"
  ],
  "tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ],
  "scheduledAuditName": "string"
}
```

## cli-input-json fields

| Name       | Type                                                      | Description                                                                                                                                                                                                                                                                                   |
|------------|-----------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| frequency  | string                                                    | How often the scheduled audit takes place. Can be one of DAILY, WEEKLY, BIWEEKLY, or MONTHLY. The actual start time of each audit is determined by the system.<br><br>enum: DAILY   WEEKLY   BIWEEKLY   MONTHLY                                                                               |
| dayOfMonth | string<br><br>pattern: ^([1-9] 1[2][0-9] 3[01])\$ ^LAST\$ | The day of the month on which the scheduled audit takes place. Can be 1 through 31 or LAST. This field is required if the frequency parameter is set to MONTHLY. If days 29-31 are specified, and the month does not have that many days, the audit takes place on the LAST day of the month. |
| dayOfWeek  | string                                                    | The day of the week on which the scheduled audit takes place. Can be one of SUN, MON, TUE, WED, THU, FRI, or SAT. This field is required if the                                                                                                                                               |

| Name               | Type                                                               | Description                                                                                                                                                                                                                                                                                                    |
|--------------------|--------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                    |                                                                    | frequency parameter is set to WEEKLY or BIWEEKLY.<br><br>enum: SUN   MON   TUE   WED   THU   FRI   SAT                                                                                                                                                                                                         |
| targetCheckNames   | list<br><br>member: AuditCheckName                                 | Which checks are performed during the scheduled audit. Checks must be enabled for your account. (Use <a href="#">DescribeAccountAuditConfiguration</a> to see the list of all checks, including those that are enabled or <a href="#">UpdateAccountAuditConfiguration</a> to select which checks are enabled.) |
| tags               | list<br><br>member: Tag<br><br>java class: java.util.List          | Metadata that can be used to manage the scheduled audit.                                                                                                                                                                                                                                                       |
| Key                | string                                                             | The tag's key.                                                                                                                                                                                                                                                                                                 |
| Value              | string                                                             | The tag's value.                                                                                                                                                                                                                                                                                               |
| scheduledAuditName | string<br><br>length- max:128 min:1<br><br>pattern: [a-zA-Z0-9_-]+ | The name you want to give to the scheduled audit. (Maximum of 128 characters)                                                                                                                                                                                                                                  |

## Output

```
{
  "scheduledAuditArn": "string"
}
```

## CLI output fields

| Name              | Type   | Description                     |
|-------------------|--------|---------------------------------|
| scheduledAuditArn | string | The ARN of the scheduled audit. |

## Errors

`InvalidRequestException`

The contents of the request were invalid.

`ThrottlingException`

The rate exceeds the limit.

#### **InternalFailureException**

An unexpected error has occurred.

#### **LimitExceededException**

A limit has been exceeded.

## ListScheduledAudits

Lists all of your scheduled audits.

### Synopsis

```
aws iot list-scheduled-audits \
[--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 for the next set of results.                                  |
| maxResults | integer<br>range- max:250 min:1 | The maximum number of results to return at one time. The default is 25. |

### Output

```
{
  "scheduledAudits": [
    {
      "scheduledAuditName": "string",
      "scheduledAuditArn": "string",
      "frequency": "string",
      "dayOfMonth": "string",
      "dayOfWeek": "string"
    }
  ],
  "nextToken": "string"
}
```

### CLI output fields

| Name            | Type | Description                   |
|-----------------|------|-------------------------------|
| scheduledAudits | list | The list of scheduled audits. |

| Name               | Type                                                                | Description                                                                                                                                                                                                          |
|--------------------|---------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                    | member:<br>ScheduledAuditMetadata<br><br>java class: java.util.List |                                                                                                                                                                                                                      |
| scheduledAuditName | string<br><br>length- max:128 min:1<br><br>pattern: [a-zA-Z0-9_-]+  | The name of the scheduled audit.                                                                                                                                                                                     |
| scheduledAuditArn  | string                                                              | The ARN of the scheduled audit.                                                                                                                                                                                      |
| frequency          | string                                                              | How often the scheduled audit takes place.<br><br>enum: DAILY   WEEKLY   BIWEEKLY   MONTHLY                                                                                                                          |
| dayOfMonth         | string<br><br>pattern: ^([1-9] [12][0-9] 3[01])\$ ^LAST\$           | The day of the month on which the scheduled audit is run (if the frequency is MONTHLY). If days 29-31 are specified, and the month does not have that many days, the audit takes place on the LAST day of the month. |
| dayOfWeek          | string                                                              | The day of the week on which the scheduled audit is run (if the frequency is WEEKLY or BIWEEKLY).<br><br>enum: SUN   MON   TUE   WED   THU   FRI   SAT                                                               |
| nextToken          | string                                                              | A token that can be used to retrieve the next set of results, or null if there are no more results.                                                                                                                  |

## Errors

### InvalidRequestException

The contents of the request were invalid.

### ThrottlingException

The rate exceeds the limit.

### InternalFailureException

An unexpected error has occurred.

## DescribeScheduledAudit

Gets information about a scheduled audit.

## Synopsis

```
aws iot describe-scheduled-audit \
--scheduled-audit-name <value> \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

**cli-input-json** format

```
{
  "scheduledAuditName": "string"
}
```

### cli-input-json fields

| Name               | Type                                                       | Description                                                        |
|--------------------|------------------------------------------------------------|--------------------------------------------------------------------|
| scheduledAuditName | string<br>length- max:128 min:1<br>pattern: [a-zA-Z0-9_-]+ | The name of the scheduled audit whose information you want to get. |

## Output

```
{
  "frequency": "string",
  "dayOfMonth": "string",
  "dayOfWeek": "string",
  "targetCheckNames": [
    "string"
  ],
  "scheduledAuditName": "string",
  "scheduledAuditArn": "string"
}
```

### CLI output fields

| Name       | Type                                                    | Description                                                                                                                                                                                                              |
|------------|---------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| frequency  | string                                                  | How often the scheduled audit takes place. One of DAILY, WEEKLY, BIWEEKLY, or MONTHLY. The actual start time of each audit is determined by the system.<br><br>enum: DAILY   WEEKLY   BIWEEKLY   MONTHLY                 |
| dayOfMonth | string<br>pattern: ^([1-9] [12][0-9] 3[01])\\$ ^LAST\\$ | The day of the month on which the scheduled audit takes place. Can be 1 through 31 or LAST. If days 29-31 are specified, and the month does not have that many days, the audit takes place on the LAST day of the month. |

| Name               | Type                                                       | Description                                                                                                                                                                                                                                                                                                        |
|--------------------|------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dayOfWeek          | string                                                     | The day of the week on which the scheduled audit takes place. One of SUN, MON, TUE, WED, THU, FRI, or SAT.<br><br>enum: SUN   MON   TUE   WED   THU   FRI   SAT                                                                                                                                                    |
| targetCheckNames   | list<br>member: AuditCheckName                             | Which checks are performed during the scheduled audit. Checks must be enabled for your account. (Use <a href="#">DescribeAccountAuditConfiguration</a> to see the list of all checks, including those that are enabled or use <a href="#">UpdateAccountAuditConfiguration</a> to select which checks are enabled.) |
| scheduledAuditName | string<br>length- max:128 min:1<br>pattern: [a-zA-Z0-9_-]+ | The name of the scheduled audit.                                                                                                                                                                                                                                                                                   |
| scheduledAuditArn  | string                                                     | The ARN of the scheduled audit.                                                                                                                                                                                                                                                                                    |

## Errors

### InvalidRequestException

The contents of the request were invalid.

### ResourceNotFoundException

The specified resource does not exist.

### ThrottlingException

The rate exceeds the limit.

### InternalFailureException

An unexpected error has occurred.

## UpdateScheduledAudit

Updates a scheduled audit, including which checks are performed and how often the audit takes place.

### Synopsis

```
aws iot update-scheduled-audit \
[--frequency <value>] \
[--day-of-month <value>] \
[--day-of-week <value>] \
[--target-check-names <value>] \
```

```
--scheduled-audit-name <value> \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

#### cli-input-json format

```
{
  "frequency": "string",
  "dayOfMonth": "string",
  "dayOfWeek": "string",
  "targetCheckNames": [
    "string"
  ],
  "scheduledAuditName": "string"
}
```

#### cli-input-json fields

| Name             | Type                                                      | Description                                                                                                                                                                                                                                                                                   |
|------------------|-----------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| frequency        | string                                                    | How often the scheduled audit takes place. Can be one of DAILY, WEEKLY, BIWEEKLY, or MONTHLY. The actual start time of each audit is determined by the system.<br><br>enum: DAILY   WEEKLY   BIWEEKLY   MONTHLY                                                                               |
| dayOfMonth       | string<br><br>pattern: ^([1-9] [12][0-9] 3[01])\$ ^LAST\$ | The day of the month on which the scheduled audit takes place. Can be 1 through 31 or LAST. This field is required if the frequency parameter is set to MONTHLY. If days 29-31 are specified, and the month does not have that many days, the audit takes place on the LAST day of the month. |
| dayOfWeek        | string                                                    | The day of the week on which the scheduled audit takes place. Can be one of SUN, MON, TUE, WED, THU, FRI, or SAT. This field is required if the frequency parameter is set to WEEKLY or BIWEEKLY.<br><br>enum: SUN   MON   TUE   WED   THU   FRI   SAT                                        |
| targetCheckNames | list<br><br>member: AuditCheckName                        | Which checks are performed during the scheduled audit. Checks must be enabled for your account. (Use <a href="#">DescribeAccountAuditConfiguration</a> to see the list of all checks, including those                                                                                         |

| Name               | Type                                                       | Description                                                                                               |
|--------------------|------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
|                    |                                                            | that are enabled or use <code>UpdateAccountAuditConfiguration</code> to select which checks are enabled.) |
| scheduledAuditName | string<br>length- max:128 min:1<br>pattern: [a-zA-Z0-9_-]+ | The name of the scheduled audit. (Maximum of 128 characters)                                              |

### Output

```
{
  "scheduledAuditArn": "string"
}
```

### CLI output fields

| Name              | Type   | Description                     |
|-------------------|--------|---------------------------------|
| scheduledAuditArn | string | The ARN of the scheduled audit. |

### Errors

`InvalidRequestException`

The contents of the request were invalid.

`ResourceNotFoundException`

The specified resource does not exist.

`ThrottlingException`

The rate exceeds the limit.

`InternalFailureException`

An unexpected error has occurred.

## DeleteScheduledAudit

Deletes a scheduled audit.

### Synopsis

```
aws iot delete-scheduled-audit \
--scheduled-audit-name <value> \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format

```
{
```

```

    "scheduledAuditName": "string"
}
```

#### **cli-input-json fields**

| Name               | Type                                                               | Description                                         |
|--------------------|--------------------------------------------------------------------|-----------------------------------------------------|
| scheduledAuditName | string<br><br>length- max:128 min:1<br><br>pattern: [a-zA-Z0-9_-]+ | The name of the scheduled audit you want to delete. |

#### **Output**

None

#### **Errors**

`InvalidRequestException`

The contents of the request were invalid.

`ResourceNotFoundException`

The specified resource does not exist.

`ThrottlingException`

The rate exceeds the limit.

`InternalFailureException`

An unexpected error has occurred.

## Run an On-Demand Audit

Use `StartOnDemandAuditTask` to specify the checks you want to perform and start an audit running right away.

### StartOnDemandAuditTask

Starts an on-demand Device Defender audit.

#### **Synopsis**

```
aws iot start-on-demand-audit-task \
--target-check-names <value> \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

#### **cli-input-json format**

```
{
  "targetCheckNames": [
    "string"
  ]
}
```

### cli-input-json fields

| Name             | Type                           | Description                                                                                                                                                                                                                                                                                                                                   |
|------------------|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| targetCheckNames | list<br>member: AuditCheckName | Which checks are performed during the audit. The checks you specify must be enabled for your account or an exception occurs. Use <a href="#">DescribeAccountAuditConfiguration</a> to see the list of all checks, including those that are enabled or use <a href="#">UpdateAccountAuditConfiguration</a> to select which checks are enabled. |

### Output

```
{
  "taskId": "string"
}
```

### CLI output fields

| Name   | Type                                                     | Description                                |
|--------|----------------------------------------------------------|--------------------------------------------|
| taskId | string<br>length- max:40 min:1<br>pattern: [a-zA-Z0-9-]+ | The ID of the on-demand audit you started. |

### Errors

`InvalidRequestException`

The contents of the request were invalid.

`ThrottlingException`

The rate exceeds the limit.

`InternalFailureException`

An unexpected error has occurred.

`LimitExceededException`

A limit has been exceeded.

## Manage Audit Instances

Use `DescribeAuditTask` to get information about a specific audit instance. If it has already run, the results include which checks failed and which passed, those that the system was unable to complete, and if the audit is still in progress, those it is still working on.

Use `ListAuditTasks` to find the audits that were run during a specified time interval.

Use `CancelAuditTask` to halt an audit in progress.

## DescribeAuditTask

Gets information about a Device Defender audit.

### Synopsis

```
aws iot describe-audit-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<br><br>length- max:40 min:1<br><br>pattern: [a-zA-Z0-9-]+ | The ID of the audit whose information you want to get. |

### Output

```
{
  "taskStatus": "string",
  "taskType": "string",
  "taskStartTime": "timestamp",
  "taskStatistics": {
    "totalChecks": "integer",
    "inProgressChecks": "integer",
    "waitingForDataCollectionChecks": "integer",
    "compliantChecks": "integer",
    "nonCompliantChecks": "integer",
    "failedChecks": "integer",
    "canceledChecks": "integer"
  },
  "scheduledAuditName": "string",
  "auditDetails": {
    "string": {
      "checkRunStatus": "string",
      "checkCompliant": "boolean",
      "totalResourcesCount": "long",
      "nonCompliantResourcesCount": "long",
      "errorCode": "string",
      "message": "string"
    }
  }
}
```

### CLI output fields

| Name                           | Type                                                               | Description                                                                                                                           |
|--------------------------------|--------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| taskStatus                     | string                                                             | The status of the audit: one of IN_PROGRESS, COMPLETED, FAILED, or CANCELED.<br><br>enum: IN_PROGRESS   COMPLETED   FAILED   CANCELED |
| taskType                       | string                                                             | The type of audit: ON_DEMAND_AUDIT_TASK or SCHEDULED_AUDIT_TASK.<br><br>enum:<br>ON_DEMAND_AUDIT_TASK   SCHEDULED_AUDIT_TASK          |
| taskStartTime                  | timestamp                                                          | The time the audit started.                                                                                                           |
| taskStatistics                 | TaskStatistics                                                     | Statistical information about the audit.                                                                                              |
| totalChecks                    | integer                                                            | The number of checks in this audit.                                                                                                   |
| inProgressChecks               | integer                                                            | The number of checks in progress.                                                                                                     |
| waitingForDataCollectionChecks | integer                                                            | The number of checks waiting for data collection.                                                                                     |
| compliantChecks                | integer                                                            | The number of checks that found compliant resources.                                                                                  |
| nonCompliantChecks             | integer                                                            | The number of checks that found noncompliant resources.                                                                               |
| failedChecks                   | integer                                                            | The number of checks.                                                                                                                 |
| canceledChecks                 | integer                                                            | The number of checks that did not run because the audit was canceled.                                                                 |
| scheduledAuditName             | string<br><br>length- max:128 min:1<br><br>pattern: [a-zA-Z0-9_-]+ | The name of the scheduled audit (only if the audit was a scheduled audit).                                                            |
| auditDetails                   | map                                                                | Detailed information about each check performed during this audit.                                                                    |
| checkRunStatus                 | string                                                             | The completion status of this check, one of IN_PROGRESS, WAITING_FOR_DATA_COLLECTION, CANCELED, COMPLETED_COMPLIANT,                  |

| Name                       | Type                       | Description                                                                                                                                                    |
|----------------------------|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                            |                            | COMPLETED_NON_COMPLIANT, or FAILED.<br><br>enum: IN_PROGRESS   WAITING_FOR_DATA_COLLECTION   CANCELED   COMPLETED_COMPLIANT   COMPLETED_NON_COMPLIANT   FAILED |
| checkCompliant             | boolean                    | True if the check completed and found all resources compliant.                                                                                                 |
| totalResourcesCount        | long                       | The number of resources on which the check was performed.                                                                                                      |
| nonCompliantResourcesCount | long                       | The number of resources that the check found noncompliant.                                                                                                     |
| errorCode                  | string                     | The code of any error encountered when performing this check during this audit. One of INSUFFICIENT_PERMISSIONS or AUDIT_CHECK_DISABLED.                       |
| message                    | string<br>length- max:2048 | The message associated with any error encountered when performing this check during this audit.                                                                |

## Errors

### InvalidRequestException

The contents of the request were invalid.

### ResourceNotFoundException

The specified resource does not exist.

### ThrottlingException

The rate exceeds the limit.

### InternalFailureException

An unexpected error has occurred.

## ListAuditTasks

Lists the Device Defender audits that have been performed during a given time period.

### Synopsis

```
aws iot list-audit-tasks \
--start-time <value> \
--end-time <value> \
[--task-type <value>] \
[--task-status <value>] \
```

```
[--next-token <value>] \
[--max-results <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

#### cli-input-json format

```
{
  "startTime": "timestamp",
  "endTime": "timestamp",
  "taskType": "string",
  "taskStatus": "string",
  "nextToken": "string",
  "maxResults": "integer"
}
```

#### cli-input-json fields

| Name       | Type                            | Description                                                                                                                                                                                                                                                                                                  |
|------------|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| startTime  | timestamp                       | The beginning of the time period. Audit information is retained for a limited time (180 days). Requesting a start time prior to what is retained results in an <code>InvalidRequestException</code> .                                                                                                        |
| endTime    | timestamp                       | The end of the time period.                                                                                                                                                                                                                                                                                  |
| taskType   | string                          | A filter to limit the output to the specified type of audit: can be one of <code>ON_DEMAND_AUDIT_TASK</code> or <code>SCHEDULED_AUDIT_TASK</code> .<br><br>enum:<br><code>ON_DEMAND_AUDIT_TASK</code>   <code>SCHEDULED_AUDIT_TASK</code>                                                                    |
| taskStatus | string                          | A filter to limit the output to audits with the specified completion status: can be one of <code>IN_PROGRESS</code> , <code>COMPLETED</code> , <code>FAILED</code> , or <code>CANCELED</code> .<br><br>enum: <code>IN_PROGRESS</code>   <code>COMPLETED</code>   <code>FAILED</code>   <code>CANCELED</code> |
| nextToken  | string                          | The token for the next set of results.                                                                                                                                                                                                                                                                       |
| maxResults | integer<br>range- max:250 min:1 | The maximum number of results to return at one time. The default is 25.                                                                                                                                                                                                                                      |

#### Output

```
{
  "tasks": [
    {
      "taskId": "string",
      "taskStatus": "string",
      "taskType": "string"
    }
  ],
  "nextToken": "string"
}
```

### CLI output fields

| Name       | Type                                                                    | Description                                                                                                                              |
|------------|-------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| tasks      | list<br><br>member: AuditTaskMetadata<br><br>java class: java.util.List | The audits that were performed during the specified time period.                                                                         |
| taskId     | string<br><br>length- max:40 min:1<br><br>pattern: [a-zA-Z0-9-]+        | The ID of this audit.                                                                                                                    |
| taskStatus | string<br><br>enum: IN_PROGRESS   COMPLETED   FAILED   CANCELED         | The status of this audit: one of IN_PROGRESS, COMPLETED, FAILED, or CANCELED.<br><br>enum: IN_PROGRESS   COMPLETED   FAILED   CANCELED   |
| taskType   | string<br><br>enum:<br>ON_DEMAND_AUDIT_TASK   SCHEDULED_AUDIT_TASK      | The type of this audit: one of ON_DEMAND_AUDIT_TASK or SCHEDULED_AUDIT_TASK.<br><br>enum:<br>ON_DEMAND_AUDIT_TASK   SCHEDULED_AUDIT_TASK |
| nextToken  | string                                                                  | A token that can be used to retrieve the next set of results, or null if there are no additional results.                                |

### Errors

**InvalidRequestException**

The contents of the request were invalid.

**ThrottlingException**

The rate exceeds the limit.

**InternalFailureException**

An unexpected error has occurred.

## CancelAuditTask

Cancels an audit that is in progress. The audit can be either scheduled or on-demand. If the audit is not in progress, an `InvalidRequestException` occurs.

### Synopsis

```
aws iot cancel-audit-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<br><br>length- max:40 min:1<br><br>pattern: [a-zA-Z0-9-]+ | The ID of the audit you want to cancel. You can only cancel an audit that is IN_PROGRESS. |

### Output

None

### Errors

`ResourceNotFoundException`

The specified resource does not exist.

`InvalidRequestException`

The contents of the request were invalid.

`ThrottlingException`

The rate exceeds the limit.

`InternalFailureException`

An unexpected error has occurred.

## Check Audit Results

Use `ListAuditFindings` to see the results of an audit. You can filter the results by the type of check, a specific resource, or the time of the audit. You can use this information to mitigate any problems that were found.

You can define mitigation actions and apply them to the findings from your audit. For more information, see [Mitigation Actions \(p. 592\)](#).

## ListAuditFindings

Lists the findings (results) of a Device Defender audit or of the audits performed during a specified time period. (Findings are retained for 180 days.)

### Synopsis

```
aws iot list-audit-findings \
[--task-id <value>] \
[--check-name <value>] \
[--resource-identifier <value>] \
[--max-results <value>] \
[--next-token <value>] \
[--start-time <value>] \
[--end-time <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

**cli-input-json** format

```
{
  "taskId": "string",
  "checkName": "string",
  "resourceIdentifier": {
    "deviceCertificateId": "string",
    "caCertificateId": "string",
    "cognitoIdentityPoolId": "string",
    "clientId": "string",
    "policyVersionIdentifier": {
      "policyName": "string",
      "policyVersionId": "string"
    },
    "roleAliasArn": "string",
    "account": "string"
  },
  "maxResults": "integer",
  "nextToken": "string",
  "startTime": "timestamp",
  "endTime": "timestamp"
}
```

**cli-input-json** fields

| Name                | Type                                                             | Description                                                                                                                                  |
|---------------------|------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| taskId              | string<br><br>length- max:40 min:1<br><br>pattern: [a-zA-Z0-9-]+ | A filter to limit results to the audit with the specified ID. You must specify either the taskId or the startTime and endTime, but not both. |
| checkName           | string                                                           | A filter to limit results to the findings for the specified audit check.                                                                     |
| resourceIdentifier  | ResourceIdentifier                                               | Information that identifies the noncompliant resource.                                                                                       |
| deviceCertificateId | string                                                           | The ID of the certificate attached to the resource.                                                                                          |

| Name                    | Type                                                          | Description                                                                                                                                        |
|-------------------------|---------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
|                         | length- max:64 min:64<br>pattern: (0x)?[a-fA-F0-9]+           |                                                                                                                                                    |
| caCertificateId         | string<br>length- max:64 min:64<br>pattern: (0x)?[a-fA-F0-9]+ | The ID of the CA certificate used to authorize the certificate.                                                                                    |
| cognitoIdentityPoolId   | string                                                        | The ID of the Amazon Cognito identity pool.                                                                                                        |
| clientId                | string                                                        | The client ID.                                                                                                                                     |
| policyVersionIdentifier | PolicyVersionIdentifier                                       | The version of the policy associated with the resource.                                                                                            |
| policyName              | string<br>length- max:128 min:1<br>pattern: [w+=,.@-]+        | The name of the policy.                                                                                                                            |
| policyVersionId         | string<br>pattern: [0-9]+                                     | The ID of the version of the policy associated with the resource.                                                                                  |
| roleAliasArn            | string                                                        | The ARN of the role alias that has overly permissive actions.<br>length- max:2048 min:1                                                            |
| account                 | string<br>length- max:12 min:12<br>pattern: [0-9]+            | The account with which the resource is associated.                                                                                                 |
| maxResults              | integer<br>range- max:250 min:1                               | The maximum number of results to return at one time. The default is 25.                                                                            |
| nextToken               | string                                                        | The token for the next set of results.                                                                                                             |
| startTime               | timestamp                                                     | A filter to limit results to those found after the specified time. You must specify either the startTime and endTime or the taskId, but not both.  |
| endTime                 | timestamp                                                     | A filter to limit results to those found before the specified time. You must specify either the startTime and endTime or the taskId, but not both. |

## Output

```
{  
  "findings": [  
    {  
      "taskId": "string",  
      "checkName": "string",  
      "taskStartTime": "timestamp",  
      "findingTime": "timestamp",  
      "severity": "string",  
      "nonCompliantResource": {  
        "resourceType": "string",  
        "resourceIdentifier": {  
          "deviceCertificateId": "string",  
          "caCertificateId": "string",  
          "cognitoIdentityPoolId": "string",  
          "clientId": "string",  
          "policyVersionIdentifier": {  
            "policyName": "string",  
            "policyVersionId": "string"  
          },  
          "account": "string"  
        },  
        "additionalInfo": {  
          "string": "string"  
        }  
      },  
      "relatedResources": [  
        {  
          "resourceType": "string",  
          "resourceIdentifier": {  
            "deviceCertificateId": "string",  
            "caCertificateId": "string",  
            "cognitoIdentityPoolId": "string",  
            "clientId": "string",  
  
            "iamRoleArn": "string",  
  
            "policyVersionIdentifier": {  
              "policyName": "string",  
              "policyVersionId": "string"  
            },  
            "account": "string"  
          },  
  
          "roleAliasArn": "string",  
  
          "additionalInfo": {  
            "string": "string"  
          }  
        }  
      ],  
      "reasonForNonCompliance": "string",  
      "reasonForNonComplianceCode": "string"  
    }  
  ],  
  "nextToken": "string"  
}
```

## CLI output fields

| Name                  | Type                                                                  | Description                                                                                                                                                 |
|-----------------------|-----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| findings              | list<br><br>member: AuditFinding                                      | The findings (results) of the audit.                                                                                                                        |
| taskId                | string<br><br>length- max:40 min:1<br><br>pattern: [a-zA-Z0-9-]+      | The ID of the audit that generated this result (finding).                                                                                                   |
| checkName             | string                                                                | The audit check that generated this result.                                                                                                                 |
| taskStartTime         | timestamp                                                             | The time the audit started.                                                                                                                                 |
| findingTime           | timestamp                                                             | The time the result (finding) was discovered.                                                                                                               |
| severity              | string                                                                | The severity of the result (finding).<br><br>enum: CRITICAL   HIGH   MEDIUM   LOW                                                                           |
| nonCompliantResource  | NonCompliantResource                                                  | The resource that was found to be noncompliant with the audit check.                                                                                        |
| resourceType          | string                                                                | The type of the noncompliant resource.<br><br>enum: DEVICE_CERTIFICATE   CA_CERTIFICATE   IOT_POLICY   COGNITO_IDENTITY_POOL   CLIENT_ID   ACCOUNT_SETTINGS |
| resourceIdentifier    | ResourceIdentifier                                                    | Information that identifies the noncompliant resource.                                                                                                      |
| deviceCertificateId   | string<br><br>length- max:64 min:64<br><br>pattern: (0x)?[a-fA-F0-9]+ | The ID of the certificate attached to the resource.                                                                                                         |
| caCertificateId       | string<br><br>length- max:64 min:64<br><br>pattern: (0x)?[a-fA-F0-9]+ | The ID of the CA certificate used to authorize the certificate.                                                                                             |
| cognitoIdentityPoolId | string                                                                | The ID of the Amazon Cognito identity pool.                                                                                                                 |
| clientId              | string                                                                | The client ID.                                                                                                                                              |

| Name                    | Type                                                                                                                        | Description                                                       |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|
| policyVersionIdentifier | PolicyVersionIdentifier                                                                                                     | The version of the policy associated with the resource.           |
| policyName              | string<br><br>length- max:128 min:1<br><br>pattern: [w+=,.@-]+                                                              | The name of the policy.                                           |
| policyVersionId         | string<br><br>pattern: [0-9]+                                                                                               | The ID of the version of the policy associated with the resource. |
| account                 | string<br><br>length- max:12 min:12<br><br>pattern: [0-9]+                                                                  | The account with which the resource is associated.                |
| additionalInfo          | map                                                                                                                         | Other information about the noncompliant resource.                |
| relatedResources        | list<br><br>member: RelatedResource                                                                                         | The list of related resources.                                    |
| resourceType            | string<br><br>enum: DEVICE_CERTIFICATE   CA_CERTIFICATE   IOT_POLICY   COGNITO_IDENTITY_POOL   CLIENT_ID   ACCOUNT_SETTINGS | The type of resource.                                             |
| resourceIdentifier      | ResourceIdentifier                                                                                                          | Information that identifies the resource.                         |
| deviceCertificateId     | string<br><br>length- max:64 min:64<br><br>pattern: (0x)?[a-fA-F0-9]+                                                       | The ID of the certificate attached to the resource.               |
| caCertificateId         | string<br><br>length- max:64 min:64<br><br>pattern: (0x)?[a-fA-F0-9]+                                                       | The ID of the CA certificate used to authorize the certificate.   |
| cognitoIdentityPoolId   | string                                                                                                                      | The ID of the Amazon Cognito identity pool.                       |
| clientId                | string                                                                                                                      | The client ID.                                                    |
| policyVersionIdentifier | PolicyVersionIdentifier                                                                                                     | The version of the policy associated with the resource.           |

| Name                       | Type                                                           | Description                                                                                                            |
|----------------------------|----------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| iamRoleArn                 | string<br><br>length- max:2048 min:20                          | The ARN of the IAM role that has overly permissive actions.                                                            |
| policyName                 | string<br><br>length- max:128 min:1<br><br>pattern: [w+=,.@-]+ | The name of the policy.                                                                                                |
| policyVersionId            | string<br><br>pattern: [0-9]+                                  | The ID of the version of the policy associated with the resource.                                                      |
| roleAliasArn               | string<br><br>length- max:2048 min:1                           | The ARN of the role alias that has overly permissive actions.                                                          |
| account                    | string<br><br>length- max:12 min:12<br><br>pattern: [0-9]+     | The account with which the resource is associated.                                                                     |
| additionalInfo             | map                                                            | Other information about the resource.                                                                                  |
| reasonForNonCompliance     | string                                                         | The reason the resource was noncompliant.                                                                              |
| reasonForNonComplianceCode | string                                                         | A code that indicates the reason that the resource was noncompliant.                                                   |
| nextToken                  | string                                                         | A token that can be used to retrieve the next set of results, or <code>null</code> if there are no additional results. |

### Errors

`InvalidRequestException`

The contents of the request were invalid.

`ThrottlingException`

The rate exceeds the limit.

`InternalFailureException`

An unexpected error has occurred.

## Mitigation Actions

You can use AWS IoT Device Defender to take actions to mitigate issues that were found during an audit. AWS IoT Device Defender provides predefined actions for the different audit checks. You configure those actions for your AWS account and then apply them to a set of findings. Those findings can be:

- All findings from an audit. This option is available in both the AWS IoT console and by using the AWS CLI.
- A list of individual findings. This option is only available by using the AWS CLI.
- A filtered set of findings from an audit.

The following table lists the types of audit checks and the supported mitigation actions for each:

#### Audit Check to Mitigation Action Mapping

| Audit Check                                          | Supported Mitigation Actions                                                       |
|------------------------------------------------------|------------------------------------------------------------------------------------|
| REVOKED_CA_CERT_CHECK                                | PUBLISH_FINDING_TO_SNS,<br>UPDATE_CA_CERTIFICATE                                   |
| DEVICE_CERTIFICATE_SHARED_CHECK                      | PUBLISH_FINDING_TO_SNS,<br>UPDATE_DEVICE_CERTIFICATE,<br>ADD_THINGS_TO_THING_GROUP |
| UNAUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK | PUBLISH_FINDING_TO_SNS                                                             |
| AUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK   | PUBLISH_FINDING_TO_SNS                                                             |
| IOT_POLICY_OVERLY_PERMISSIVE_CHECK                   | PUBLISH_FINDING_TO_SNS,<br>REPLACE_DEFAULT_POLICY_VERSION                          |
| CA_CERT_APPROACHING_EXPIRATION_CHECK                 | PUBLISH_FINDING_TO_SNS,<br>UPDATE_CA_CERTIFICATE                                   |
| CONFLICTING_CLIENT_IDS_CHECK                         | PUBLISH_FINDING_TO_SNS                                                             |
| DEVICE_CERT_APPROACHING_EXPIRATION_CHECK             | PUBLISH_FINDING_TO_SNS,<br>UPDATE_DEVICE_CERTIFICATE,<br>ADD_THINGS_TO_THING_GROUP |
| REVOKED_DEVICE_CERT_CHECK                            | PUBLISH_FINDING_TO_SNS,<br>UPDATE_DEVICE_CERTIFICATE,<br>ADD_THINGS_TO_THING_GROUP |
| LOGGING_DISABLED_CHECK                               | PUBLISH_FINDING_TO_SNS,<br>ENABLE_IOT_LOGGING                                      |
| DEVICE_CERTIFICATE_KEY_QUALITY_CHECK                 | PUBLISH_FINDING_TO_SNS,<br>UPDATE_DEVICE_CERTIFICATE,<br>ADD_THINGS_TO_THING_GROUP |
| CA_CERTIFICATE_KEY_QUALITY_CHECK                     | PUBLISH_FINDING_TO_SNS,<br>UPDATE_CA_CERTIFICATE                                   |
| IOT_ROLE_ALIAS_OVERLY_PERMISSIVE_CHECK               | PUBLISH_FINDING_TO_SNS                                                             |
| IOT_ROLE_ALIAS_ALLows_ACCESS_TO_UNUSED_S             | PUBLISH_FINDING_TO_SNS                                                             |

All audit checks support publishing the audit findings to Amazon SNS so you can take custom actions in response to the notification. Each type of audit check can support additional mitigation actions:

#### REVOKED\_CA\_CERT\_CHECK

- Change the state of the certificate to mark it as inactive in AWS IoT.

#### **DEVICE\_CERTIFICATE\_SHARED\_CHECK**

- Change the state of the device certificate to mark it as inactive in AWS IoT.
- Add the devices that use that certificate to a thing group.

#### **UNAUTHENTICATED\_COGNITO\_ROLE\_OVERLY\_PERMISSIVE\_CHECK**

- No additional supported actions.

#### **AUTHENTICATED\_COGNITO\_ROLE\_OVERLY\_PERMISSIVE\_CHECK**

- No additional supported actions.

#### **IOT\_POLICY\_OVERLY\_PERMISSIVE\_CHECK**

- Add a blank AWS IoT policy version to restrict permissions.

#### **CA\_CERT\_APPROACHING\_EXPIRATION\_CHECK**

- Change the state of the certificate to mark it as inactive in AWS IoT.

#### **CONFLICTING\_CLIENT\_IDS\_CHECK**

- No additional supported actions.

#### **DEVICE\_CERT\_APPROACHING\_EXPIRATION\_CHECK**

- Change the state of the device certificate to mark it as inactive in AWS IoT.
- Add the devices that use that certificate to a thing group.

#### **DEVICE\_CERTIFICATE\_KEY\_QUALITY\_CHECK**

- Change the state of the device certificate to mark it as inactive in AWS IoT.
- Add the devices that use that certificate to a thing group.

#### **CA\_CERTIFICATE\_KEY\_QUALITY\_CHECK**

- Change the state of the certificate to mark it as inactive in AWS IoT.

#### **REVOKED\_DEVICE\_CERT\_CHECK**

- Change the state of the device certificate to mark it as inactive in AWS IoT.
- Add the devices that use that certificate to a thing group.

#### **LOGGING\_DISABLED\_CHECK**

- Enable logging.

AWS IoT Device Defender supports the following types of mitigation actions:

| Action type                    | Notes                                                                                                                                                                                                                               |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ADD_THINGS_TO_THING_GROUP      | You specify the group to which you want to add the devices. You also specify whether membership in one or more dynamic groups should be overridden if that would exceed the maximum number of groups to which the thing can belong. |
| ENABLE_IOT_LOGGING             | You specify the logging level and the role with permissions for logging. You cannot specify a logging level of DISABLED.                                                                                                            |
| PUBLISH_FINDING_TO_SNS         | You specify the topic to which the finding should be published.                                                                                                                                                                     |
| REPLACE_DEFAULT_POLICY_VERSION | You specify the template name. Replaces the policy version with a default or blank policy. Only a value of BLANK_POLICY is currently supported.                                                                                     |

| Action type               | Notes                                                                                                                 |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------|
| UPDATE_CA_CERTIFICATE     | You specify the new state for the CA certificate. Only a value of <code>DEACTIVATE</code> is currently supported.     |
| UPDATE_DEVICE_CERTIFICATE | You specify the new state for the device certificate. Only a value of <code>DEACTIVATE</code> is currently supported. |

By configuring standard actions when issues are found during an audit, you can respond to those issues consistently. Using these defined mitigation actions also helps you resolve the issues more quickly and with less chance of human error.

**Important**

Applying mitigation actions that change certificates, add things to a new thing group, or replace the policy can have an impact on your devices and applications. For example, devices might be unable to connect. Consider the implications of the mitigation actions before you apply them. You might need to take other actions to correct the problems before your devices and applications can function normally. For example, you might need to provide updated device certificates. Mitigation actions can help you quickly limit your risk, but you must still take corrective actions to address the underlying issues.

Some actions, such as reactivating a device certificate, can only be performed manually. AWS IoT Device Defender does not provide a mechanism to automatically roll back mitigation actions that have been applied.

## How to Define and Manage Mitigation Actions

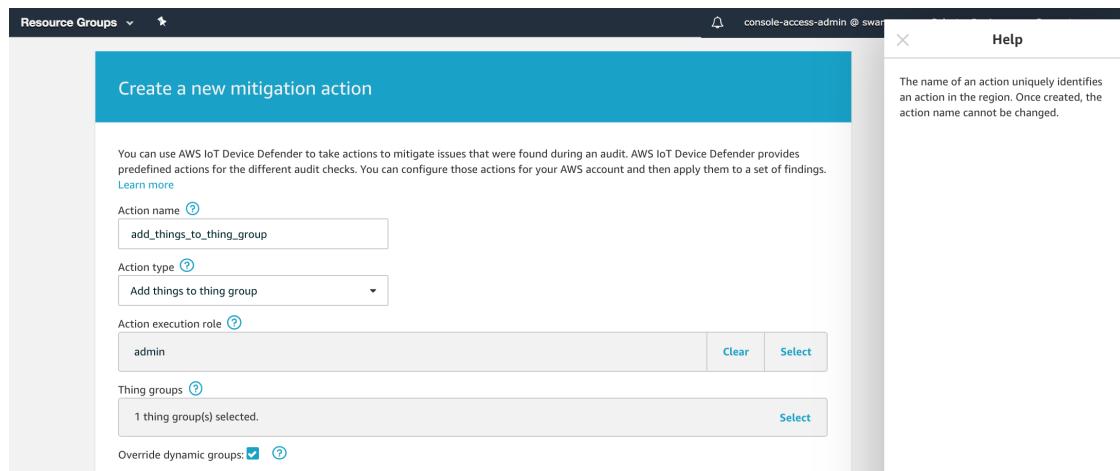
You can use the AWS IoT console or the AWS CLI to define and manage mitigation actions for your AWS account.

### Create Mitigation Actions

Each mitigation action that you define is a combination of a predefined action type and parameters specific to your account.

#### To use the AWS IoT console to create mitigation actions

1. Open the [AWS IoT console](#).
2. In the left navigation pane, choose **Defend**, and then choose **Mitigation Actions**.
3. On the **Mitigation Actions** page, choose **Create**.



4. On the **Create a Mitigation Action** page, in **Action name**, enter a unique name for your mitigation action.
5. In **Action type**, specify the type of action that you want to define.
6. Each action type requests a different set of parameters. Enter the parameters for the action. For example, if you choose the **Add things to thing group** action type, choose the destination group and select or clear **Override dynamic groups**.
7. In **Action execution role**, choose the role under whose permissions the action is applied.
8. Choose **Save** to save your mitigation action to your AWS account.

### To use the AWS CLI to create mitigation actions

- Use the [CreateMitigationAction \(p. 607\)](#) command to create your mitigation action. The unique name that you give the action is used when you apply that action to audit findings. Choose a meaningful name.

### To use the AWS IoT console to view and modify mitigation actions

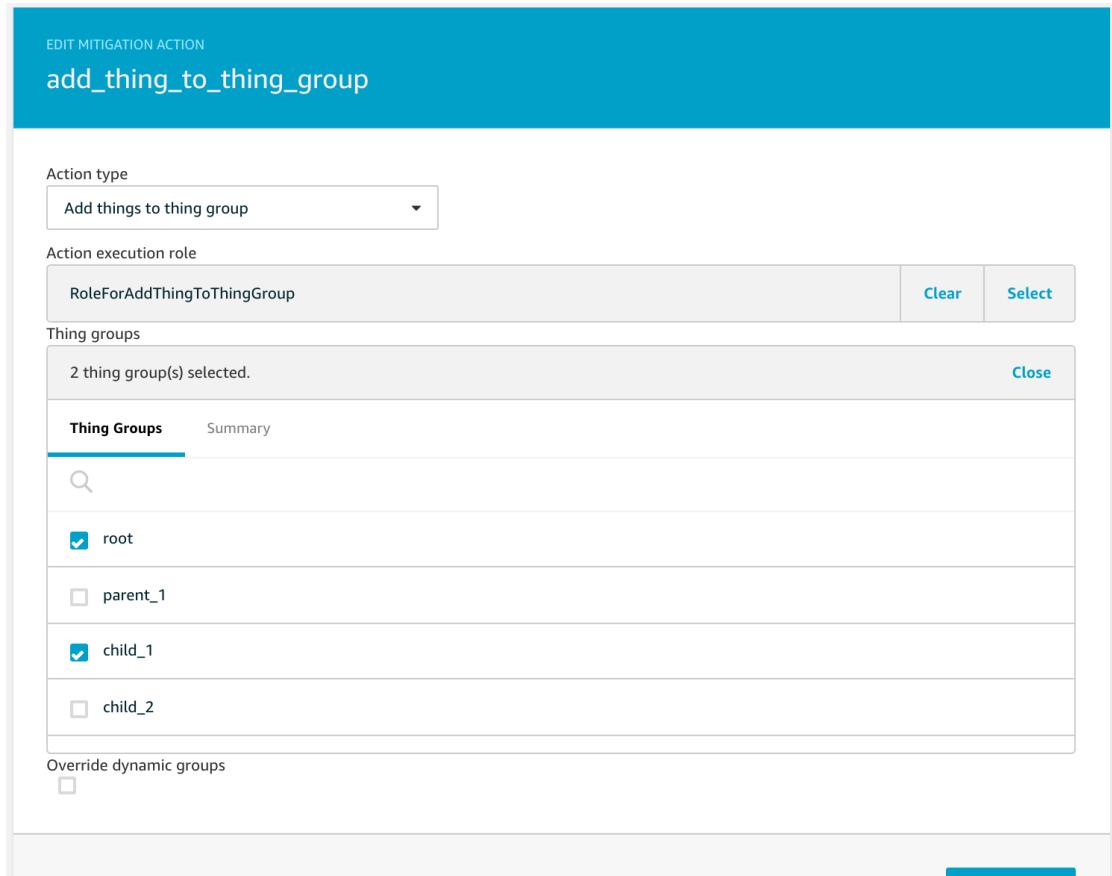
1. Open the [AWS IoT console](#).
2. In the left navigation pane, choose **Defend**, and then choose **Mitigation Actions**.

The **Mitigation Actions** page displays a list of all of the mitigation actions that are defined for your AWS account.

| Mitigation actions (4)         |                                |                                                 |                      |
|--------------------------------|--------------------------------|-------------------------------------------------|----------------------|
| Created date                   | Action name                    | ARN                                             |                      |
| Jun 10, 2019 10:09:53 AM -0700 | enable_logging                 | arn:aws:iot:us-east-1:xxxxxx:mitigationa... *** | <a href="#">More</a> |
| Jun 6, 2019 6:08:47 PM -0700   | sns_publish                    | arn:aws:iot:us-east-1:xxxxxx:mitigationa... *** | <a href="#">More</a> |
| Jun 6, 2019 6:08:26 PM -0700   | replace_default_policy_version | arn:aws:iot:us-east-1:xxxxxx:mitigationa... *** | <a href="#">More</a> |
| Jun 3, 2019 10:51:16 PM -0700  | add_thing_to_thing_group       | arn:aws:iot:us-east-1:xxxxxx:mitigationa... *** | <a href="#">More</a> |

3. Choose the action name link for the mitigation action that you want to change.

4. Make your changes to the mitigation action. Because the name of the mitigation action is used to identify it, you cannot change the name.



5. Choose **Save** to save the changes to the mitigation action to your AWS account.

#### To use the AWS CLI to list a mitigation action

- Use the [ListMitigationActions \(p. 614\)](#) command to list your mitigation actions. If you want to change or delete a mitigation action, make a note of the name.

#### To use the AWS CLI to update a mitigation action

- Use the [UpdateMitigationAction \(p. 611\)](#) command to change your mitigation action.

#### To use the AWS IoT console to delete a mitigation action

1. Open the [AWS IoT console](#).
2. In the left navigation pane, choose **Defend**, and then choose **Mitigation Actions**.

The **Mitigation Actions** page displays all of the mitigation actions that are defined for your AWS account.

3. Choose the ellipsis (...) for the mitigation action that you want to delete, and then choose **Delete**.

### To use the AWS CLI to delete mitigation actions

- Use the [UpdateMitigationAction \(p. 611\)](#) command to change your mitigation action.

### To use the AWS IoT console to view mitigation action details

1. Open the [AWS IoT console](#).
2. In the left navigation pane, choose **Defend**, and then choose **Mitigation Actions**.

The screenshot shows the 'Mitigation actions (4)' section of the AWS IoT Device Defender console. At the top, there are buttons for 'Create' and a refresh icon. Below is a table with four rows of data:

| Created date                   | Action name                    | ARN                                          | More |
|--------------------------------|--------------------------------|----------------------------------------------|------|
| Jun 10, 2019 10:09:53 AM -0700 | enable_logging                 | arn:aws:iot:us-east-1:...:mitigationa... *** |      |
| Jun 6, 2019 6:08:47 PM -0700   | sns_publish                    | arn:aws:iot:us-east-1:...:mitigationa... *** |      |
| Jun 6, 2019 6:08:26 PM -0700   | replace_default_policy_version | arn:aws:iot:us-east-1:...:mitigationa... *** |      |
| Jun 3, 2019 10:51:16 PM -0700  | add_thing_to_thing_group       | arn:aws:iot:us-east-1:...:mitigationa... *** |      |

The **Mitigation Actions** page displays all of the mitigation actions that are defined for your AWS account.

3. Choose the action name link for the mitigation action that you want to change.
4. In the **Are you sure you want to delete the mitigation action** window, choose **Confirm**.

The dialog box has a blue header bar with the text 'Are you sure you want to delete the mitigation action?'. The main body contains the message: 'You will no longer be able to use this action to mitigate non-compliant resources identified by AWS IoT Device Defender.' At the bottom, there are 'Cancel' and 'Confirm' buttons, and a status bar at the bottom says 'Role: admin'.

### To use the AWS CLI to view mitigation action details

- Use the [DescribeMitigationAction \(p. 616\)](#) command to view details for your mitigation action.

## Apply Mitigation Actions

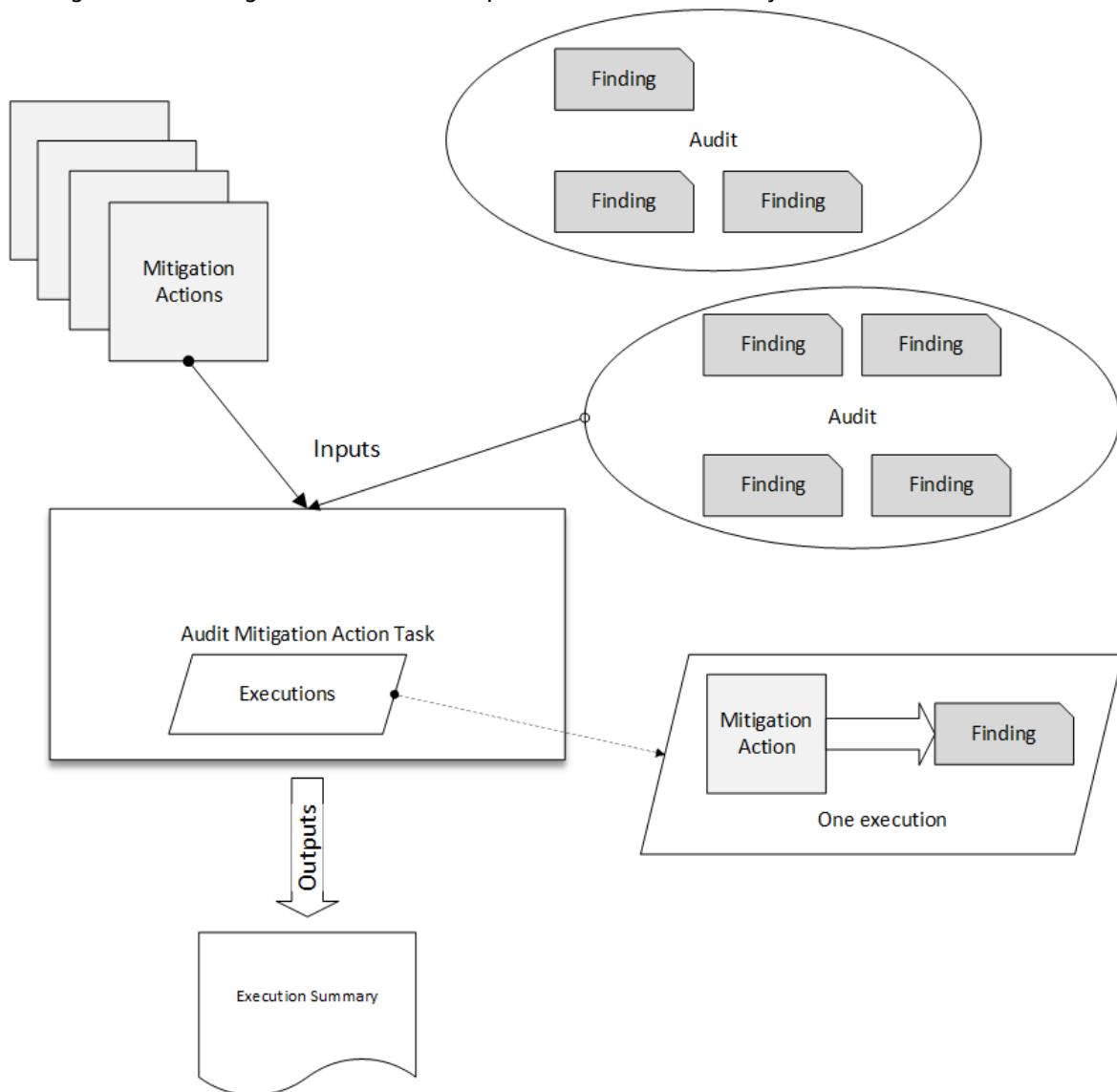
After you have defined a set of mitigation actions, you can apply those actions to the findings from an audit. When you apply actions, you start an audit mitigation actions task. This task might take some time to complete, depending on the set of findings and the actions that you apply to them. For example, if you have a large pool of devices whose certificates have expired, it might take some time to deactivate

all of those certificates or to move those devices to a quarantine group. Other actions, such as enabling logging, can be completed quickly.

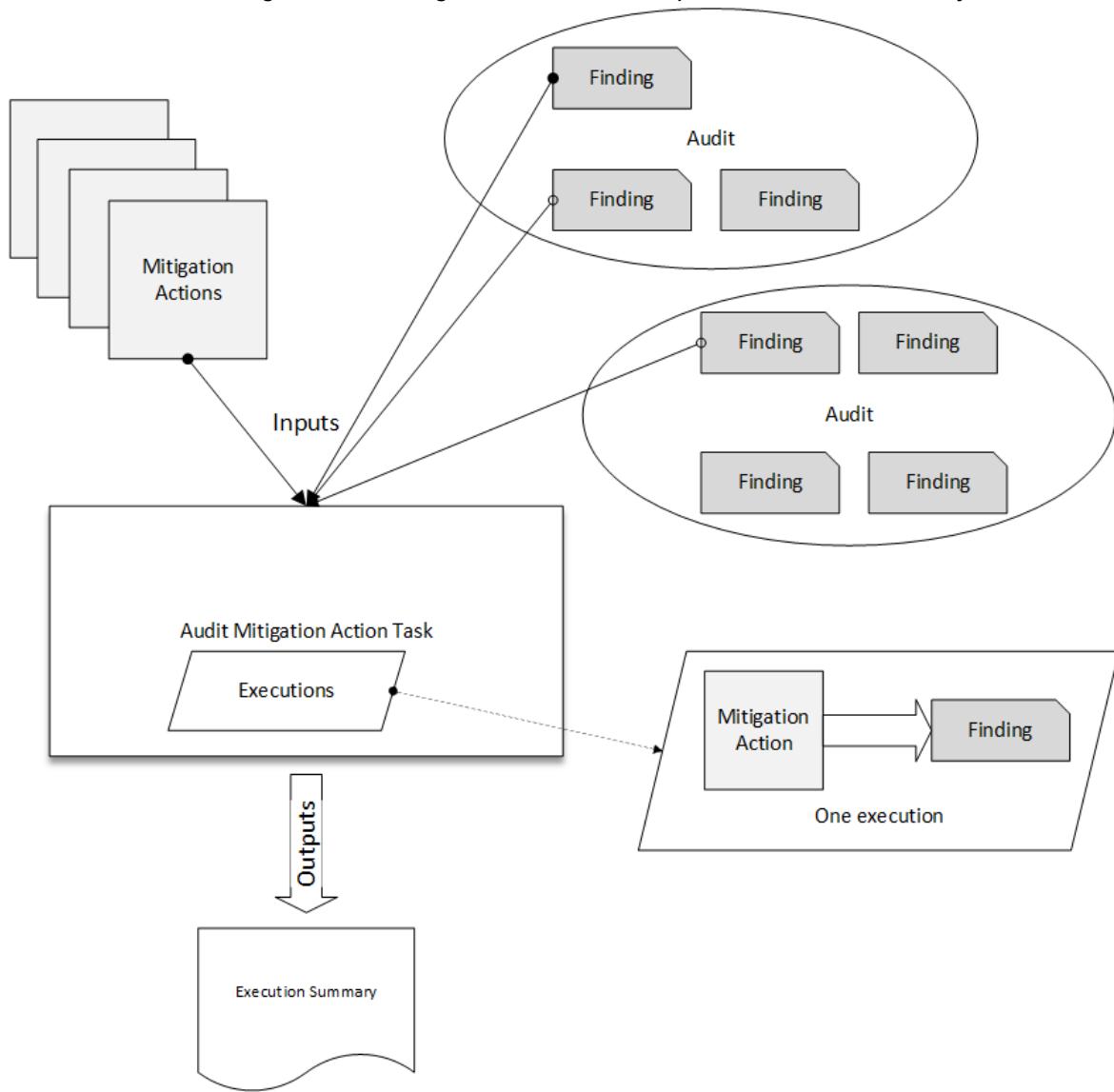
You can view the list of action executions and cancel an execution that has not yet been completed. Actions already performed as part of the canceled action execution are not rolled back. If you are applying multiple actions to a set of findings and one of those actions failed, the subsequent actions are skipped for that finding (but are still applied to other findings). The task status for the finding is FAILED. The taskStatus is set to failed if one or more of the actions failed when applied to the findings. Actions are applied in the order in which they are specified.

Each action execution applies a set of actions to a target. That target can be a list of findings or it can be all findings from an audit.

The following diagram shows how you can define an audit mitigation task that takes all findings from one audit and applies a set of actions to those findings. A single execution applies one action to one finding. The audit mitigation actions task outputs an execution summary.



The following diagram shows how you can define an audit mitigation task that takes a list of individual findings from one or more audits and applies a set of actions to those findings. A single execution applies one action to one finding. The audit mitigation actions task outputs an execution summary.



You can use the AWS IoT console or the AWS CLI to apply mitigation actions.

#### To use the AWS IoT console to apply mitigation actions by starting an action execution

1. Open the [AWS IoT console](#).
2. In the left navigation pane, choose **Defend**, choose **Audit**, and then choose **Results**.

The screenshot shows the AWS IoT Device Defender Audit Results page. At the top, it displays the path: Device Defender > Audit > Results > On-demand. Below this, the audit report title is "On-demand - Jun 9, 2019 10:26:36 PM -0700". A blue button on the right says "Start mitigation actions". The main content area is titled "Audit findings" and shows an audit task ID and start time. It lists "Non-compliant checks (5 of 10)" with the following details:

| Check name                                                | Severity | Non-compliant | % Resources | Mitigation                                   |
|-----------------------------------------------------------|----------|---------------|-------------|----------------------------------------------|
| CA certificate revoked but device certificates still a... | Critical | 79            | 83.2%       | Review & deactivate <a href="#">(1)</a>      |
| CA certificate expiring                                   | Medium   | 79            | 83.2%       | Reprovision & deactivate <a href="#">(1)</a> |
| Device certificate expiring                               | Medium   | 50            | 92.6%       | Reprovision & deactivate <a href="#">(1)</a> |
| Revoked device certificate still active                   | Medium   | 50            | 92.6%       | Reprovision & revoke <a href="#">(1)</a>     |
| Logging disabled                                          | Low      | 1             | 100%        | Enable logging <a href="#">(1)</a>           |

**3. Choose the name for the audit to which you want to apply actions.**

The screenshot shows the "Start a new mitigation action" dialog box. It has a header with "Resource Groups" and a "Help" link. The main area contains the following fields:

- Task name:** A text input field containing the audit ID: "8115481b332374e3309c13050320323c".
- Select options for Device certificate expiring:**
  - Select actions:** A section showing "1 actions selected" with an "Expand" button.
  - Select reason codes:** A section showing "2 reason codes selected" with an "Expand" button.
- Buttons:** "Cancel" and "Confirm".

To the right of the dialog, there is a note: "Each mitigation action task is identified by a unique user-provided name. AWS IoT Device Defender Audit does not run multiple executions of a task with the same name."

4. Choose **Start Mitigation Actions**. This button is not available if all of your checks are compliant.
5. In **Are you sure that you want to start mitigation action task**, the task name defaults to the audit ID, but you can change it to something more meaningful.
6. For each type of check that had one or more noncompliant findings in the audit, you can choose one or more actions to apply. Only actions that are valid for the check type are displayed.

**Note**

If you have not configured actions for your AWS account, the list of actions is empty. You can choose the **click here** link to create one or more mitigation actions.

7. When you have specified all of the actions that you want to apply, choose **Confirm**.

**To use the AWS CLI to apply mitigation actions by starting an audit mitigation actions execution**

1. If you want to apply actions to all findings for the audit, use the [ListAuditTasks \(p. 583\)](#) command to find the task ID.

2. If you want to apply actions to selected findings only, use the [ListAuditFindings \(p. 587\)](#) command to get the finding IDs.
3. Use the [ListMitigationActions \(p. 614\)](#) command and make note of the names of the mitigation actions that you want to apply.
4. Use the [StartAuditMitigationActionsTask \(p. 621\)](#) command to apply actions to the target. Make note of the task ID. You can use the ID to check the state of the action execution, review the details, or cancel it.

### To use the AWS IoT console to view your action executions

1. Open the [AWS IoT console](#).
2. In the left navigation pane, choose **Defend**, and then choose **Action Executions**.

The screenshot shows the AWS IoT Device Defender Action Executions page. The URL is `Device Defender > Audit > Action executions`. The page title is "Action tasks (1)". Below the title, it says "1-1 of 1". There is a table with three columns: "Date", "Name", and "Status". The first row shows a task started on "Jun 6, 2019 6:09:07 PM -0700" with the name "[ff82164a6439e6024e83b4fc104817d7](#)" and status "Completed".

A list of action tasks shows when each was started and the current status.

3. Choose the **Name** link to see details for the task. The details include all of the actions that are applied by the task, their target, and their status.

The screenshot shows the "MITIGATION ACTION EXECUTION TASK" details for task `ff82164a6439e6024e83b4fc104817d7`. The URL is `Device Defender > Audit > Action executions > ff82164a6439e6024e83b4fc104817d7`. The task status is "COMPLETED". The "Details" section shows the task was started at "Jun 6, 2019 6:09:07 PM -0700" and completed at "Jun 6, 2019 6:09:09 PM -0700". The "Check summary" section shows a table with columns: Check name, Failed, Successful, Skipped, Canceled, Total, and Executions. For the check "IoT policies overly permissive", the values are: Failed 0, Successful 2, Skipped 0, Canceled 0, Total 2, and Executions 2. A "Show" button is also present.

You can use the **Show executions for** filters to focus on types of actions or action states.

4. To see details for the task, in **Executions**, choose **Show**.

The screenshot shows the AWS Device Defender Audit Action executions page. At the top, it displays the path: Device Defender > Audit > Action executions > ff82164a6439e6024e83b4fc104817d7 >. Below this, a dark header bar contains the text "MITIGATION ACTION EXECUTION TASK" and the ID "ff82164a6439e6024e83b4fc104817d7". The main content area has a heading "IoT policies overly permissive". Underneath, there's a section titled "Action executions (4)" with a "Show executions for" dropdown set to "All actions" and "All status". A table follows, showing 1-4 of 4 rows of execution details:

| Started at                   | Status    | Action                         | Finding                       |
|------------------------------|-----------|--------------------------------|-------------------------------|
| Jun 6, 2019 6:09:08 PM -0700 | Completed | sns_publish                    | 053cff17-1da4-4479-996b-8b... |
| Jun 6, 2019 6:09:08 PM -0700 | Completed | replace_default_policy_version | 053cff17-1da4-4479-996b-8b... |
| Jun 6, 2019 6:09:08 PM -0700 | Completed | replace_default_policy_version | 2b966f76-b499-4986-836c-f8... |

### To use the AWS CLI to list your started tasks

1. Use [ListAuditMitigationActionsTasks \(p. 627\)](#) to view your audit mitigation actions tasks. You can provide filters to narrow the results. If you want to view details of the task, make note of the task ID.
2. Use [ListAuditMitigationActionsExecutions \(p. 624\)](#) to view execution details for a particular audit mitigation actions task.
3. Use [DescribeAuditMitigationActionsTask \(p. 630\)](#) to view details about the task, such as the parameters specified when it was started.

### To use the AWS CLI to cancel a running audit mitigation actions task

1. Use the [ListAuditMitigationActionsTasks \(p. 627\)](#) command to find the task ID for the task whose execution you want to cancel. You can provide filters to narrow the results.
2. Use the [CancelAuditMitigationActionsTask \(p. 624\)](#) command, using the task ID, to cancel your audit mitigation actions task. You cannot cancel tasks that have been completed. When you cancel a task, remaining actions are not applied, but mitigation actions that were already applied are not rolled back.

## Permissions

For each mitigation action that you define, you must provide the role used to apply that action.

### Permissions for Mitigation Actions

| Action Type               | Permissions Policy Template                                         |
|---------------------------|---------------------------------------------------------------------|
| UPDATE_DEVICE_CERTIFICATE | {         "Version": "2012-10-17",         "Statement": [         { |

| Action Type               | Permissions Policy Template                                                                                                                                                                                                                                            |  |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
|                           | <pre>     "Effect": "Allow",       "Action": [         "iot:UpdateCertificate"       ],       "Resource": [         "*"       ]     } } </pre>                                                                                                                         |  |
| UPDATE_CA_CERTIFICATE     | <pre> {   "Version": "2012-10-17",   "Statement": [     {       "Effect": "Allow",         "Action": [           "iot:UpdateCACertificate"         ],         "Resource": [           "*"         ]       }     ] } </pre>                                             |  |
| ADD_THINGS_TO_THING_GROUP | <pre> {   "Version": "2012-10-17",   "Statement": [     {       "Effect": "Allow",         "Action": [           "iot&gt;ListPrincipalThings",           "iot&gt;AddThingToThingGroup"         ],         "Resource": [           "*"         ]       }     ] } </pre> |  |

| Action Type                    | Permissions Policy Template                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| REPLACE_DEFAULT_POLICY_VERSION | <pre>{     "Version": "2012-10-17",     "Statement": [         {             "Effect": "Allow",             "Action": [                 "iot:CreatePolicyVersion"             ],             "Resource": [                 "*"             ]         }     ] }</pre>                                                                                                                                                                                                                                      |
| ENABLE_IOT_LOGGING             | <pre>{     "Version": "2012-10-17",     "Statement": [         {             "Effect": "Allow",             "Action": [                 "iot:SetV2LoggingOptions"             ],             "Resource": [                 "*"             ]         },         {             "Effect": "Allow",             "Action": [                 "iam:PassRole"             ],             "Resource": [                 "&lt;IAM role ARN used for setting up logging&gt;"             ]         }     ] }</pre> |

| Action Type            | Permissions Policy Template                                                                                                                                                                                                                                                                                    |  |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| PUBLISH_FINDING_TO_SNS | <pre>{     "Version": "2012-10-17",     "Statement": [         {             "Effect": "Allow",             "Action": [                 "sns:Publish"             ],             "Resource": [                 "&lt;The SNS topic to which the finding is published&gt;"             ]         }     ] }</pre> |  |

For all mitigation action types, use the following trust policy template:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

## Mitigation Action Commands

You use these mitigation action commands to define a set of actions for your AWS account that you can later apply to one or more sets of audit findings. There are two command categories:

- Those used to define and manage actions.
- Those used to start and manage the application of those actions to audit findings.

### Mitigation Action Commands

| Define and Manage Actions                         | Start and Manage Execution                                  |
|---------------------------------------------------|-------------------------------------------------------------|
| <a href="#">CreateMitigationAction (p. 607)</a>   | <a href="#">CancelAuditMitigationActionsTask (p. 624)</a>   |
| <a href="#">DeleteMitigationAction (p. 620)</a>   | <a href="#">DescribeAuditMitigationActionsTask (p. 630)</a> |
| <a href="#">DescribeMitigationAction (p. 616)</a> | <a href="#">ListAuditMitigationActionsTasks (p. 627)</a>    |

| Define and Manage Actions                       | Start and Manage Execution                                    |
|-------------------------------------------------|---------------------------------------------------------------|
| <a href="#">ListMitigationActions (p. 614)</a>  | <a href="#">StartAuditMitigationActionsTask (p. 621)</a>      |
| <a href="#">UpdateMitigationAction (p. 611)</a> | <a href="#">ListAuditMitigationActionsExecutions (p. 624)</a> |

## CreateMitigationAction

Defines an action that can be applied to audit findings by using [StartAuditMitigationActionsTask \(p. 621\)](#). Each mitigation action can apply only one type of change. Defining an action does not apply it.

### Synopsis

```
aws iot create-mitigation-action \
  --action-name <value> \
  --role-arn <value> \
  [--tags <value>] \
  --action-params <value> \
  [--cli-input-json <value>] \
  [--generate-cli-skeleton]
```

cli-input-json format

```
{
  "actionParams": {
    "addThingsToThingGroupParams": {
      "overrideDynamicGroups": boolean,
      "thingGroupNames": [ "string" ]
    },
    "enableIoTLoggingParams": {
      "logLevel": "string",
      "roleArnForLogging": "string"
    },
    "publishFindingToSnsParams": {
      "topicArn": "string"
    },
    "replaceDefaultPolicyVersionParams": {
      "templateName": "string"
    },
    "updateCACertificateParams": {
      "action": "string"
    },
    "updateDeviceCertificateParams": {
      "action": "string"
    }
  },
  "roleArn": "string",
  "tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ]
}
```

### cli-input-json fields

| Name         | Type                              | Description                                                                                                                                                        |
|--------------|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| roleArn      | string<br>length- max:2048 min:20 | The ARN of the role that grants permission to AWS IoT to access information about your devices, policies, certificates, and other items when applying this action. |
| tags         | array of tag objects              | Metadata that can be used to manage the mitigation action.                                                                                                         |
| actionParams | map                               | Defines the type of action to be applied and the parameters for that mitigation action. You can include only one type of parameter for each mitigation action.     |

You must provide parameters for the type of action that you are defining. You can provide only one action type and its parameters. These are the supported action types:

- ADD\_THINGS\_TO\_THING\_GROUP
- ENABLE\_IOT\_LOGGING
- PUBLISH\_FINDING\_TO\_SNS
- REPLACE\_DEFAULT\_POLICY\_VERSION
- UPDATE\_CA\_CERTIFICATE
- UPDATE\_DEVICE\_CERTIFICATE

### Parameters for AddThingsToThingGroup

| Name                  | Type             | Description                                                                                                                                                                                                                                                                            |
|-----------------------|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| overrideDynamicGroups | boolean          | Optional. Specifies if this mitigation action can move the things that triggered the mitigation action out of one or more dynamic thing groups. This setting is used only if the thing is already in the maximum number of groups.                                                     |
| thingGroupNames       | array of strings | Required. The list of groups to which you want to add the things that triggered the mitigation action.<br><br>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.<br><br>You must provide at least one group name. |

| Name | Type | Description                                                                                     |
|------|------|-------------------------------------------------------------------------------------------------|
|      |      | Length constraints: Minimum length of 1. Maximum length of 128.<br><br>Pattern: [a-zA-Z0-9:_-]+ |

#### Parameters for EnableIoTLogging

| Name              | Type   | Description                                                                                                                                                                                   |
|-------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| logLevel          | string | Required. Specifies which types of information are logged.<br><br>Valid values, from most verbose to least, are DEBUG, INFO, ERROR, and WARN. You cannot specify a logging level of DISABLED. |
| roleArnForLogging | string | Required. The ARN of the IAM role used for logging.<br><br>Minimum length of 20.<br>Maximum length of 2048.                                                                                   |

#### Parameters for PublishingFindingToSns

| Name     | Type   | Description                                                                                                                       |
|----------|--------|-----------------------------------------------------------------------------------------------------------------------------------|
| topicArn | string | Required. The ARN of the topic to which you want to publish the findings.<br><br>Minimum length of 20.<br>Maximum length of 2048. |

#### Parameters for ReplaceDefaultPolicyVersion

| Name         | Type   | Description                                                                                        |
|--------------|--------|----------------------------------------------------------------------------------------------------|
| templateName | string | Required. The name of the template to be applied.<br><br>The only supported value is BLANK_POLICY. |

#### Parameters for UpdateCACertificate

| Name   | Type   | Description                                                        |
|--------|--------|--------------------------------------------------------------------|
| action | string | Required. The action that you want to apply to the CA certificate. |

| Name | Type | Description                             |
|------|------|-----------------------------------------|
|      |      | The only supported value is DEACTIVATE. |

### Parameters for UpdateDeviceCertificate

| Name   | Type   | Description                                                                                                           |
|--------|--------|-----------------------------------------------------------------------------------------------------------------------|
| action | string | Required. The action that you want to apply to the device certificate.<br><br>The only supported value is DEACTIVATE. |

### CLI output fields:

| Name      | Type   | Description                                          |
|-----------|--------|------------------------------------------------------|
| actionArn | string | The ARN for the new mitigation action.               |
| actionId  | string | The unique identifier for the new mitigation action. |

### Errors

#### InvalidRequestException

The contents of the request were invalid.

#### LimitExceeded**Exception**

A limit has been exceeded. For information about mitigation action limits, see [Service Limits](#).

#### RequestAlreadyExists**Exception**

A mitigation action with this name already exists. This error occurs only if another mitigation action exists with the same name but different parameters.

#### Throttling**Exception**

The rate exceeds the limit.

#### InternalFailure**Exception**

An unexpected error has occurred.

### Example

This example defines a mitigation action that, when applied, deactivates a device certificate.

```
$ aws iot create-mitigation-action --action-name "UpdateCACertName" --role-arn arn:aws:iam::123456789012:role/MitigationActionsValidRole --action-params "updateDeviceCertificateParams={action=DEACTIVATE}"
```

The response resembles the following:

```
{
    "actionArn": "arn:aws:iot:us-east-1:123456789012:mitigationaction/UpdateCACertName",
    "actionId": "6a22b98e-0e27-4396-9b25-637d04959429"
}
```

## UpdateMitigationAction

Updates the definition of an action that can be applied to audit findings by using [StartAuditMitigationActionsTask \(p. 621\)](#). Each mitigation action can apply only one type of change.

### Synopsis

```
aws iot update-mitigation-action \
--action-name <value> \
--role-arn <value> \
--action-params <value> \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

**cli-input-json** format

```
{
    "actionParams": {
        "addThingsToThingGroupParams": {
            "overrideDynamicGroups": boolean,
            "thingGroupNames": [ "string" ]
        },
        "enableIoTLoggingParams": {
            "logLevel": "string",
            "roleArnForLogging": "string"
        },
        "publishFindingToSnsParams": {
            "topicArn": "string"
        },
        "replaceDefaultPolicyVersionParams": {
            "templateName": "string"
        },
        "updateCACertificateParams": {
            "action": "string"
        },
        "updateDeviceCertificateParams": {
            "action": "string"
        }
    },
    "roleArn": "string"
}
```

**cli-input-json** fields

| Name    | Type                              | Description                                                                                     |
|---------|-----------------------------------|-------------------------------------------------------------------------------------------------|
| roleArn | string<br>length- max:2048 min:20 | The ARN of the role that grants permission to AWS IoT to access information about your devices, |

| Name         | Type | Description                                                                                                                                         |
|--------------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
|              |      | policies, certificates, and other items when applying this action.                                                                                  |
| actionParams | map  | Defines the type of action to be applied and the parameters for that mitigation action. You can specify only one type of action and its parameters. |

You must provide parameters for the type of action that you are defining. You can provide only one action type and its parameters. These are the supported action types:

- ADD\_THINGS\_TO\_THING\_GROUP
- ENABLE\_IOT\_LOGGING
- PUBLISH\_FINDING\_TO\_SNS
- REPLACE\_DEFAULT\_POLICY\_VERSION
- UPDATE\_CA\_CERTIFICATE
- UPDATE\_DEVICE\_CERTIFICATE

#### Parameters for AddThingsToThingGroup

| Name                  | Type             | Description                                                                                                                                                                                                                                                                                                                                                                                          |
|-----------------------|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| overrideDynamicGroups | boolean          | Optional. Specifies if this mitigation action can move the things that triggered the mitigation action out of one or more dynamic thing groups. This setting is used only if the thing is already in the maximum number of groups.                                                                                                                                                                   |
| thingGroupNames       | array of strings | <p>Required. The list of groups to which you want to add the things that triggered the mitigation action.</p> <p>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.</p> <p>You must provide at least one group name.</p> <p>Length constraints: Minimum length of 1. Maximum length of 128.</p> <p>Pattern: [a-zA-Z0-9:_-]+</p> |

### Parameters for EnableIoTLogging

| Name              | Type   | Description                                                                                                                                                                                               |
|-------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| logLevel          | string | <p>Required. Specifies what types of information are to be logged.</p> <p>Valid values, from most verbose to least, are DEBUG, INFO, ERROR, and WARN. You cannot specify a logging level of DISABLED.</p> |
| roleArnForLogging | string | <p>Required. The ARN of the IAM role used for logging.</p> <p>Minimum length of 20.<br/>Maximum length of 2048.</p>                                                                                       |

### Parameters for PublishingFindingToSns

| Name     | Type   | Description                                                                                                                               |
|----------|--------|-------------------------------------------------------------------------------------------------------------------------------------------|
| topicArn | string | <p>Required. The ARN of the topic to which you want to publish the findings.</p> <p>Minimum length of 20.<br/>Maximum length of 2048.</p> |

### Parameters for ReplaceDefaultPolicyVersion

| Name         | Type   | Description                                                                                               |
|--------------|--------|-----------------------------------------------------------------------------------------------------------|
| templateName | string | <p>Required. The name of the template to be applied.</p> <p>The only supported value is BLANK_POLICY.</p> |

### Parameters for UpdateCACertificate

| Name   | Type   | Description                                                                                                              |
|--------|--------|--------------------------------------------------------------------------------------------------------------------------|
| action | string | <p>Required. The action that you want to apply to the CA certificate.</p> <p>The only supported value is DEACTIVATE.</p> |

### Parameters for UpdateDeviceCertificate

| Name   | Type   | Description                                                                                                                         |
|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------|
| action | string | <p>Required. The action that you want to apply to the device certificate.</p> <p>The only supported value is <b>DEACTIVATE</b>.</p> |

### cli output fields:

| Name      | Type   | Description                                      |
|-----------|--------|--------------------------------------------------|
| actionArn | string | The ARN for the mitigation action.               |
| actionId  | string | The unique identifier for the mitigation action. |

### Errors

**InvalidRequestException**

The contents of the request were invalid.

**ResourceNotFoundException**

No mitigation action with the specified name was found.

**ThrottlingException**

The rate exceeds the limit.

**InternalFailureException**

An unexpected error has occurred.

## ListMitigationActions

You use the **ListMitigationActions** command to get a list of all mitigation actions in your AWS account.

### Synopsis

```
aws iot list-mitigation-actions \
[--action-type <value>] \
[--max-results <value>] \
[--next-token <value>]
```

### Input parameters

| Name        | Type   | Description                                                                             |
|-------------|--------|-----------------------------------------------------------------------------------------|
| action-type | string | Specify a value to limit the results to mitigation actions with a specific action type. |

| Name        | Type    | Description                                                                                                                                                                      |
|-------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |         | Supported values are UPDATE_DEVICE_CERTIFICATE, UPDATE_CA_CERTIFICATE, ADD_THINGS_TO_THING_GROUP, REPLACE_DEFAULT_POLICY_VERSION, ENABLE_IOT_LOGGING, or PUBLISH_FINDING_TO_SNS. |
| max-results | integer | The maximum number of results to return at one time.<br><br>The default is 25. Valid range is from 1 to 250.                                                                     |
| next-token  | string  | The token to retrieve the next set of results.                                                                                                                                   |

Output JSON:

```
{
  "actionIdentifiers": [
    {
      "actionArn": "string",
      "actionName": "string",
      "creationDate": number
    }
  ],
  "nextToken": "string"
}
```

#### CLI output fields:

| Name              | Type      | Description                                                                                                       |
|-------------------|-----------|-------------------------------------------------------------------------------------------------------------------|
| actionIdentifiers | map       | Information about the collection of mitigation actions that match the specified parameters.                       |
| actionArn         | string    | The ARN for the mitigation action.                                                                                |
| actionName        | string    | The unique identifier for the new mitigation action.<br><br>Maximum length is 128.<br><br>Pattern: [a-zA-Z0-9_-]+ |
| creationDate      | timestamp | The date and time when the mitigation action was created.                                                         |
| nextToken         | string    | The token to retrieve the next set of results.                                                                    |

## Errors

`InvalidRequestException`

The contents of the request were invalid.

`ThrottlingException`

The rate exceeds the limit.

`InternalFailureException`

An unexpected error has occurred.

## Example

This example lists all mitigation actions defined for your account in the region.

```
$ aws iot list-mitigation-actions
```

The response resembles the following:

```
{
    "actionIdentifiers": [
        {
            "actionName": "UpdateCACertName",
            "actionArn": "arn:aws:iot:us-east-1:123456789012:mitigationaction/
UpdateCACertName",
            "creationDate": 1560173584.521
        }
    ]
}
```

## DescribeMitigationAction

You use the **DescribeMitigationAction** command to view details for a mitigation action in your AWS account.

### Synopsis

```
aws iot describe-mitigation-action --action-name <value>
```

### Input parameters

| Name                     | Type   | Description                                                                                                                    |
|--------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------|
| <code>action-name</code> | string | The friendly name that uniquely identifies the mitigation action.<br><br>Maximum length is 128.<br><br>Pattern: [a-zA-Z0-9_-]+ |

Output JSON:

```
{
    "actionArn": "string",
    "actionId": "string",
    "actionName": "string",
    "actionParams": {
        "addThingsToThingGroupParams": {
            "overrideDynamicGroups": boolean,
            "thingGroupNames": [ "string" ]
        },
        "enableIoTLoggingParams": {
            "logLevel": "string",
            "roleArnForLogging": "string"
        },
        "publishFindingToSnsParams": {
            "topicArn": "string"
        },
        "replaceDefaultPolicyVersionParams": {
            "templateName": "string"
        },
        "updateCACertificateParams": {
            "action": "string"
        },
        "updateDeviceCertificateParams": {
            "action": "string"
        }
    },
    "actionType": "string",
    "creationDate": number,
    "lastModifiedDate": number,
    "roleArn": "string"
}
```

Each mitigation action has only one type, so only one of the sets of parameters appears.

### CLI output fields:

| Name                        | Type   | Description                                                                                                          |
|-----------------------------|--------|----------------------------------------------------------------------------------------------------------------------|
| actionArn                   | string | The ARN for the mitigation action.                                                                                   |
| actionId                    | string | A unique identifier for the mitigation action.                                                                       |
| actionName                  | string | The unique friendly name for the new mitigation action.<br><br>Maximum length is 128.<br><br>Pattern: [a-zA-Z0-9_-]+ |
| actionParams                | map    | Defines the type of action to be applied and the parameters for that mitigation action.                              |
| addThingsToThingGroupParams | map    | Parameters to define a mitigation action that moves devices associated with a certificate to one or more             |

| Name                      | Type             | Description                                                                                                                                                                                                                                                                                                                                                           |
|---------------------------|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           |                  | specified thing groups, typically for quarantine.                                                                                                                                                                                                                                                                                                                     |
| overrideDynamicGroups     | boolean          | Specifies if this mitigation action can move the things that triggered the mitigation action even if they are part of one or more dynamic thing groups.                                                                                                                                                                                                               |
| thingGroupNames           | array of strings | <p>The list of groups to which you want to add the things that triggered the mitigation action.</p> <p>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.</p> <p>You must provide at least one group name.</p> <p>Minimum length of 1. Maximum length of 128.</p> <p>Pattern: [a-zA-Z0-9:_]+</p> |
| enableIoTLoggingParams    | map              | Parameters to define a mitigation action that enables AWS IoT logging at a specified level of detail.                                                                                                                                                                                                                                                                 |
| logLevel                  | string           | <p>Specifies the types of information to log.</p> <p>Valid values, from most verbose to least, are DEBUG, INFO, ERROR, WARN, and DISABLED.</p>                                                                                                                                                                                                                        |
| roleArnForLogging         | string           | <p>The ARN of the IAM role used for logging.</p> <p>Minimum length of 20. Maximum length of 2048.</p>                                                                                                                                                                                                                                                                 |
| publishFindingToSnsParams | map              | Parameters to define a mitigation action that publishes findings to Amazon SNS. You can implement your own custom actions in response to the Amazon SNS messages.                                                                                                                                                                                                     |
| topicArn                  | string           | <p>The ARN of the topic to which you want to publish the findings.</p> <p>Minimum length of 20. Maximum length of 2048.</p>                                                                                                                                                                                                                                           |

| Name                              | Type      | Description                                                                                                         |
|-----------------------------------|-----------|---------------------------------------------------------------------------------------------------------------------|
| replaceDefaultPolicyVersionParams | map       | Parameters to define a mitigation action that adds a blank policy to restrict permissions.                          |
| templateName                      | string    | The name of the template to be applied.<br><br>The only supported value is <b>BLANK_POLICY</b> .                    |
| updateCACertificateParams         | map       | Parameters to define a mitigation action that changes the state of the CA certificate to inactive.                  |
| action                            | string    | The action that you want to apply to the CA certificate.<br><br>The only supported value is <b>DEACTIVATE</b> .     |
| updateDeviceCertificateParams     | map       | Parameters to define a mitigation action that changes the state of the device certificate to inactive.              |
| action                            | string    | The action that you want to apply to the device certificate.<br><br>The only supported value is <b>DEACTIVATE</b> . |
| actionType                        | string    | The type of action being applied.                                                                                   |
| creationDate                      | timestamp | The date and time when the mitigation action was created.                                                           |
| lastModifiedDate                  | timestamp | The date and time when the mitigation action was most recently changed.                                             |
| roleArn                           | string    | The ARN for the role to use when applying this mitigation action.                                                   |

## Errors

**ResourceNotFoundException**

No mitigation action with the specified name was found.

**InvalidRequestException**

The contents of the request were invalid.

### ThrottlingException

The rate exceeds the limit.

### InternalFailureException

An unexpected error has occurred.

### Example

This example gets the definition for the mitigation action named `UpdateCACertName`.

```
$ aws iot describe-mitigation-action --action-name UpdateCACertName
```

The response resembles the following:

```
{  
    "actionName": "UpdateCACertName",  
    "actionType": "UPDATE_DEVICE_CERTIFICATE",  
    "actionArn": "arn:aws:iot:us-east-1:123456789012:mitigationaction/UpdateCACertName",  
    "actionId": "6a22b98e-0e27-4396-9b25-637d04959429",  
    "roleArn": "arn:aws:iam::123456789012:role/MitigationActionsValidRole",  
    "actionParams": {  
        "updateDeviceCertificateParams": {  
            "action": "DEACTIVATE"  
        }  
    },  
    "creationDate": 1560173584.521,  
    "lastModifiedDate": 1560173584.521  
}
```

## DeleteMitigationAction

You use the **DeleteMitigationAction** command to remove a mitigation action from your AWS account. To make the **DeleteMitigationAction** command idempotent, it does not throw a `ResourceNotFoundException` if you try to delete a mitigation action that doesn't exist. Instead, **DeleteMitigationAction** returns success when the action name does not exist.

### Synopsis

```
aws iot delete-mitigation-action --action-name <value>
```

### Input parameters

| Name        | Type   | Description                                                                                                                    |
|-------------|--------|--------------------------------------------------------------------------------------------------------------------------------|
| action-name | string | The friendly name that uniquely identifies the mitigation action.<br><br>Maximum length is 128.<br><br>Pattern: [a-zA-Z0-9_-]+ |

### Errors

#### **InvalidRequestException**

The contents of the request were invalid.

#### **ThrottlingException**

The rate exceeds the limit.

#### **InternalFailureException**

An unexpected error has occurred.

## StartAuditMitigationActionsTask

You use the **StartAuditMitigationActionsTask** command to apply a set of mitigation actions to findings from one or more audits.

### Synopsis

```
aws iot start-audit-mitigation-actions-task \
--task-id <value> \
--audit-task-id <value> \
--audit-checks-to-action-mapping <value> \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

**cli-input-json** format

```
{
    "auditCheckToActionsMapping": {
        "string" : [ "string" ]
    },
    "clientRequestToken": "string",
    "target": {
        "auditTaskId": "string",
        "findingIds": [ "string" ]
    }
}
```

### Input parameters

| Name                       | Type                           | Description                                                                                                                                                                                        |
|----------------------------|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| taskId                     | string                         | A unique identifier for the audit mitigations task.<br><br>Maximum length is 128.<br><br>Pattern: [a-zA-Z0-9_-]+                                                                                   |
| auditCheckToActionsMapping | string to array of strings map | Specifies the list of actions to apply to a particular audit check.<br><br>Array must contain at least 1 but not more than 5 members.<br><br>Maximum length is 128.<br><br>Pattern: [a-zA-Z0-9_-]+ |

| Name               | Type             | Description                                                                                                                                                                                                                                                                                                                                                                |
|--------------------|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| clientRequestToken | string           | <p>Each audit mitigation task must have a unique client request token. If you try to start a new task with the same task ID as an existing task, but with a different token, an exception occurs. If you omit this value, a unique client request token is generated automatically.</p> <p>Minimum length of 1. Maximum length of 64.</p> <p>Pattern: [a-zA-Z0-9_-]+\$</p> |
| target             | map              | <p>Specifies the audit findings to which the mitigation actions are applied. You can apply them to the results of an audit task or a set of findings.</p>                                                                                                                                                                                                                  |
| auditTaskId        | string           | <p>A unique identifier for the audit to whose findings you want to apply the set of actions. The <code>auditCheckToReasonCodeFilter</code> can further filter the results from the audit. If you want to restrict the actions to a specific set of findings, use <code>findingIds</code> instead.</p> <p>Maximum length is 40.</p> <p>Pattern: [a-zA-Z0-9\_-]+</p>         |
| findingIds         | array of strings | <p>If the task applies a mitigation action to one or more listed findings, this value uniquely identifies those findings.</p> <p>Array members: Minimum of 1 item. Maximum of 25 items.</p> <p>Maximum length of each item is 128.</p> <p>Pattern: [a-zA-Z0-9_-]+</p>                                                                                                      |

Output JSON:

```
{
    "taskId": "string"
}
```

### CLI output fields:

| Name   | Type   | Description                                                                                                                                                                                                                                                                              |
|--------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| taskId | string | The unique identifier for the newly created audit mitigation actions task.<br><br>Use this identifier if you need to cancel the task ( <code>CancelAuditMitigationActionsTask</code> ) or if you want to view details of the task with <code>DescribeAuditMitigationActionsTask</code> . |

### Errors

#### TaskAlreadyExistsException

This exception occurs if you attempt to start a task with the same task-id as an existing task but with a different clientRequestToken.

#### InvalidRequestException

The contents of the request were invalid.

#### LimitExceededException

This exception occurs if you try to start more than 10 mitigation action tasks.

#### ThrottlingException

The rate exceeds the limit.

#### InternalFailureException

An unexpected error has occurred.

### Example

This example starts a task to apply the `UpdateCACertAction` that you defined to the audit findings from the audit whose taskId is `aef320b958891041e0c60c088afac64c` and the audit check is `CA_CERTIFICATE_EXPIRING_CHECK`.

```
$ aws iot start-audit-mitigation-actions-task --task-id "myActionsTaskId" --target "auditTaskId=aef320b958891041e0c60c088afac64c" --audit-check-to-actions-mapping "CA_CERTIFICATE_EXPIRING_CHECK=UpdateCACertAction" --client-request-token "adhadhahda"
```

The response looks like the following.

```
{
    "taskId": "myActionsTaskId"
}
```

## CancelAuditMitigationActionsTask

You use the **CancelAuditMitigationActionsTask** command to stop execution for an audit mitigation task if it is not already complete.

### Synopsis

```
aws iot cancel-audit-mitigation-actions-task --task-id <value>
```

### Input parameters

| Name    | Type   | Description                                                                                                                              |
|---------|--------|------------------------------------------------------------------------------------------------------------------------------------------|
| task-id | string | A unique identifier for the audit mitigations task that you want to cancel.<br><br>Maximum length is 128.<br><br>Pattern: [a-zA-Z0-9_-]+ |

### Errors

**ResourceNotFoundException**

An audit mitigation actions task with the specified task ID was not found.

**InvalidRequestException**

The contents of the request were invalid.

**ThrottlingException**

The rate exceeds the limit.

**InternalFailureException**

An unexpected error has occurred.

## ListAuditMitigationActionsExecutions

You use the **ListAuditMitigationActionsExecutions** command to display details about an audit mitigation task that has been started.

### Synopsis

```
aws iot list-audit-mitigation-actions-executions \
[--action-status <value>] \
[--finding-id <value>] \
[--max-results <value>] \
[--next-token <value>] \
[--task-id <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

### Input parameters

| Name          | Type   | Description                                                                                                                                                                   |
|---------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| action-status | string | Specify this filter to limit results to those executions with a specific status.<br><br>Supported values are IN_PROGRESS, COMPLETED, FAILED, CANCELED, SKIPPED , and PENDING. |
| finding-id    | string | Specify this filter to limit results to those that were applied to a specific audit finding.<br><br>Maximum length is 128.<br><br>Pattern: [a-zA-Z0-9_-] +                    |
| max-results   | number | The maximum number of results to return at one time. The default is 25.<br><br>Valid range: minimum of 1. maximum of 250.                                                     |
| next-token    | string | The token for the next set of results.                                                                                                                                        |
| task-id       | string | A unique identifier for the audit mitigations task whose details you want to display.<br><br>Maximum length is 128.<br><br>Pattern: [a-zA-Z0-9_-] +                           |

Output JSON:

```
{
  "actionsExecutions": [
    {
      "actionId": "string",
      "actionName": "string",
      "endTime": number,
      "errorCode": "string",
      "findingId": "string",
      "message": "string",
      "startTime": number,
      "status": "string",
      "taskId": "string"
    }
  ],
  "nextToken": "string"
}
```

**CLI output fields:**

| Name              | Type      | Description                                                                                                                         |
|-------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------|
| actionsExecutions | map       | A collection of information about the audit mitigation task executions that have been started for your AWS account.                 |
| actionId          | string    | The unique identifier for the mitigation action being applied by the task.                                                          |
| actionName        | string    | The friendly name of the mitigation action being applied by the task.                                                               |
| endTime           | timestamp | The date and time when the task was completed or canceled.                                                                          |
| errorCode         | string    | If an error occurred, the code that indicates the error type.                                                                       |
| findingId         | string    | The unique identifier for the finding to which the task and associated mitigation actions are applied.                              |
| message           | string    | If an error occurred, a message that describes the error.                                                                           |
| startTime         | timestamp | The date and time when the task was started.                                                                                        |
| status            | string    | The current status of the task being executed. Supported values are IN_PROGRESS, COMPLETED, FAILED, CANCELED, SKIPPED , and PENDING |
| taskId            | string    | The unique identifier for the task that applies the mitigation action.                                                              |
| nextToken         | string    | The token for the next set of results.                                                                                              |

The status field can have the following values:

| Status      | What It Means                                                             |
|-------------|---------------------------------------------------------------------------|
| IN_PROGRESS | AWS IoT Device Defender is applying the mitigation action to the finding. |
| COMPLETED   | The mitigation action was applied successfully to the finding.            |

| Status   | What It Means                                                                                        |
|----------|------------------------------------------------------------------------------------------------------|
| FAILED   | The mitigation action failed to be applied to the finding. The error code provides more information. |
| CANCELED | The action execution was canceled because the user canceled the task.                                |
| SKIPPED  | The action execution was skipped because one of the actions in the list failed.                      |
| PENDING  | The action execution has not started yet.                                                            |

The following error codes can be returned:

#### **INSUFFICIENT\_PERMISSIONS**

- The roleArn that was provided for the mitigation action does not have permissions to apply the action.

#### **INVALID\_STATE\_OF\_RESOURCE**

- The resource in the finding is not in a state that allows the mitigation action to be applied successfully. This error occurs for the ADD\_THINGS\_TO\_THING\_GROUP action type if the thing is already in the maximum number of allowed groups. The error occurs for the UPDATE\_DEVICE\_CERTIFICATE and UPDATE\_CA\_CERTIFICATE mitigation action types if the certificate has already been revoked.

#### **Errors**

##### **InvalidRequestException**

The contents of the request were invalid.

##### **ThrottlingException**

The rate exceeds the limit.

##### **InternalFailureException**

An unexpected error has occurred.

## ListAuditMitigationActionsTasks

You use the **ListAuditMitigationActionsTasks** command to get a list of audit mitigation action tasks that match the specified filters.

#### **Synopsis**

```
aws iot list-audit-mitigation-actions-tasks
[--audit-task-id <value>] \
--end-time <value> \
[--finding-id <value>] \
[--max-results <value>] \
[--next-token <value>] \
--start-time <value> \
[--task-status <value>] \
[--cli-input-json <value>] \
```

```
[--generate-cli-skeleton]
```

### Input parameters

| Name          | Type      | Description                                                                                                                                                          |
|---------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| audit-task-id | string    | A unique identifier for the audit mitigations task that you want to display.<br><br>Minimum length is 1. Maximum length is 40.<br><br>Pattern: [a-zA-Z0-9_-]+        |
| end-time      | timestamp | Required. Specify this filter to list tasks that ended on or before this date. This date should not be more than 90 days in the past.                                |
| finding-id    | string    | Specify this filter to list tasks that applied to a particular finding.<br><br>Maximum length is 128.<br><br>Pattern: [a-zA-Z0-9_-]+                                 |
| max-results   | number    | The maximum number of results to return at one time. The default is 25.<br><br>Valid range: Minimum value is 1. Maximum value is 250.                                |
| next-token    | string    | The token for the next set of results.                                                                                                                               |
| start-time    | timestamp | Required. Specify this filter to limit the results to tasks that were started on or after this date and time. This date should not be more than 90 days in the past. |
| task-status   | string    | Specify this filter to limit the results to tasks that are in a specified state.<br><br>Supported values are: IN_PROGRESS, COMPLETED, FAILED, and CANCELED.          |

Output JSON:

```
{
  "nextToken": "string",
```

```

    "tasks": [
      {
        "startTime": number,
        "taskId": "string",
        "taskStatus": "string"
      }
    ]
}

```

### CLI output fields:

| Name       | Type      | Description                                                                                                                                                |
|------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nextToken  | string    | The token for the next set of results.                                                                                                                     |
| tasks      | map       | The collection of audit mitigation action tasks that matched the filter criteria.                                                                          |
| startTime  | timestamp | Parameters to define a mitigation action that moves devices associated with a certificate to one or more specified thing groups, typically for quarantine. |
| taskId     | string    | The unique identifier for the task.                                                                                                                        |
| taskStatus | string    | The current state of the task.                                                                                                                             |

### Errors

#### InvalidRequestException

The contents of the request were invalid. This error can occur if you specify dates more than 90 days in the past.

#### ThrottlingException

The rate exceeds the limit.

#### InternalFailureException

An unexpected error has occurred.

### Example

This example lists the audit mitigation tasks that ran during the specified timeframe.

```
$ aws iot list-audit-mitigation-actions-tasks --start-time 1560001663 --end-time 1560174463
```

The response looks like the following.

```
{  
    "tasks": [  
        {  
            "taskId": "actionsTaskId",  
            "startTime": 1560174232.07,  
            "taskStatus": "CANCELED"  
        },  
        {  
            "taskId": "4e8acacf8-b7f0-4484-9b0d-01b979e61d8d",  
            "startTime": 1560060994.965,  
            "taskStatus": "COMPLETED"  
        },  
        {  
            "taskId": "0e8f5a95-e43f-43fa-9aa4-57ff3efb946d",  
            "startTime": 1560060860.243,  
            "taskStatus": "COMPLETED"  
        },  
        {  
            "taskId": "92d60009-33a1-45a7-a3c6-b28e938ec68a",  
            "startTime": 1560060707.653,  
            "taskStatus": "COMPLETED"  
        },  
        {  
            "taskId": "bdea63c8-58bd-4798-9cb3-e949c7f1969a",  
            "startTime": 1560060531.123,  
            "taskStatus": "COMPLETED"  
        },  
        {  
            "taskId": "61e146ef-69f4-41bf-9d37-5d667f72ef3d",  
            "startTime": 1560060388.035,  
            "taskStatus": "COMPLETED"  
        },  
        {  
            "taskId": "2ef289dc-9bbd-422b-95ba-d96b7e626a60",  
            "startTime": 1560060256.695,  
            "taskStatus": "COMPLETED"  
        },  
        {  
            "taskId": "a6cdc16b-6be3-4800-bafa-2b86d3f5d639",  
            "startTime": 1560060097.613,  
            "taskStatus": "COMPLETED"  
        },  
        {  
            "taskId": "7ccf351b-e560-4eb2-8796-1dc1bcb25396",  
            "startTime": 1560059925.477,  
            "taskStatus": "COMPLETED"  
        },  
        {  
            "taskId": "8f69ee5d-3e35-4c91-88e8-3c5f84c96fde",  
            "startTime": 1560059345.473,  
            "taskStatus": "COMPLETED"  
        }  
    ]  
}
```

## DescribeAuditMitigationActionsTask

You use the **DescribeAuditMitigationActionsTask** command to display details about an audit mitigation task that has been started.

### Synopsis

```
aws iot describe-audit-mitigation-actions-task --taskId <value>
```

### Input parameters

| Name   | Type   | Description                                                                                                                                                         |
|--------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| taskId | string | <p>Required. A unique identifier for the audit mitigations task whose details you want to display.</p> <p>Maximum length is 128.</p> <p>Pattern: [a-zA-Z0-9_-]+</p> |

Output JSON:

```
{
    "actionsDefinition": [
        {
            "actionParams": {
                "addThingsToThingGroupParams": {
                    "overrideDynamicGroups": boolean,
                    "thingGroupNames": [ "string" ]
                },
                "enableIoTLoggingParams": {
                    "logLevel": "string",
                    "roleArnForLogging": "string"
                },
                "publishFindingToSnsParams": {
                    "topicArn": "string"
                },
                "replaceDefaultPolicyVersionParams": {
                    "templateName": "string"
                },
                "updateCACertificateParams": {
                    "action": "string"
                },
                "updateDeviceCertificateParams": {
                    "action": "string"
                }
            },
            "id": "string",
            "name": "string",
            "roleArn": "string"
        }
    ],
    "auditCheckToActionsMapping": {
        "string" : [ "string" ]
    },
    "endTime": number,
    "startTime": number,
    "target": {
        "auditCheckToReasonCodeFilter": {
            "string" : [ "string" ]
        },
        "auditTaskId": "string",
        "findingIds": [ "string" ]
    },
    "taskStatistics": {
        "string" : {

```

```

        "canceledFindingsCount": number,
        "failedFindingsCount": number,
        "skippedFindingsCount": number,
        "succeededFindingsCount": number,
        "totalFindingsCount": number
    }
},
"taskStatus": "string"
}

```

**CLI output fields:**

| Name                       | Type             | Description                                                                                                                                                       |
|----------------------------|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| actionsDefinition          | map              | The set of actions (and their metadata) applied by this audit mitigation action task.                                                                             |
| actionParams               | map              | The parameters used when applying the mitigation action.                                                                                                          |
| addThingsToThingGroupParam | map              | Parameters to define a mitigation action that moves devices associated with a certificate to one or more specified thing groups, typically for quarantine.        |
| overrideDynamicGroups      | boolean          | Specifies if this mitigation action can move the things that triggered the mitigation action even if they are part of one or more dynamic thing groups.           |
| thingGroupNames            | array of strings | The list of groups to which this mitigation action adds the things in the target for this audit mitigation actions task.                                          |
| enableIoTLoggingParams     | map              | Parameters to define a mitigation action that enables AWS IoT logging at a specified level of detail.                                                             |
| logLevel                   | string           | Specifies the types of information to log.                                                                                                                        |
| roleArnForLogging          | string           | The ARN of the IAM role used for logging.                                                                                                                         |
| publishFindingToSnsParams  | map              | Parameters to define a mitigation action that publishes findings to Amazon SNS. You can implement your own custom actions in response to the Amazon SNS messages. |
| topicArn                   | string           | The ARN of the topic to which you want to publish the findings.                                                                                                   |

| Name                              | Type      | Description                                                                                                         |
|-----------------------------------|-----------|---------------------------------------------------------------------------------------------------------------------|
| replaceDefaultPolicyVersionParams | map       | Parameters to define a mitigation action that adds a blank policy to restrict permissions.                          |
| templateName                      | string    | The name of the template to be applied.<br><br>The only supported value is <b>BLANK_POLICY</b> .                    |
| updateCACertificateParams         | map       | Parameters to define a mitigation action that changes the state of the CA certificate to inactive.                  |
| action                            | string    | The action that you want to apply to the CA certificate.<br><br>The only supported value is <b>DEACTIVATE</b> .     |
| updateDeviceCertificateParams     | map       | Parameters to define a mitigation action that changes the state of the device certificate to inactive.              |
| action                            | string    | The action that you want to apply to the device certificate.<br><br>The only supported value is <b>DEACTIVATE</b> . |
| id                                | string    | The unique identifier for the mitigation action applied as part of this audit mitigation task.                      |
| name                              | string    | The friendly name for the mitigation action applied as part of this audit mitigation task.                          |
| roleArn                           | string    | The ARN for the role used when applying this mitigation task.                                                       |
| auditCheckToActionsMapping        | map       | For a type of audit check, specifies which mitigation actions are applied by this audit mitigation task.            |
| endTime                           | timestamp | If the audit mitigation action task is complete or was canceled, the date and time when execution stopped.          |
| startTime                         | timestamp | The date and time when execution of this audit mitigation action task began.                                        |

| Name                         | Type             | Description                                                                                                                                                                          |
|------------------------------|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| target                       | map              | Information about the targets to which the mitigation actions are applied as part of this audit mitigation actions task.                                                             |
| auditCheckToReasonCodeFilter | map              | Specifies a set of audit checks and reason codes used to identify the audit findings to which this audit mitigation actions task applies.                                            |
| auditTaskId                  | string           | Specifies an audit on whose findings this audit mitigation actions task applies.                                                                                                     |
| findingIds                   | array of strings | Specifies a set of audit findings on which this audit mitigation actions task applies.                                                                                               |
| taskStatistics               | map              | The set of execution statistics for this audit mitigation actions task.                                                                                                              |
| canceledFindingsCount        | number           | If the task was canceled, the number of findings in the target for this audit mitigation actions task to which mitigation actions were not applied.                                  |
| failedFindingsCount          | number           | The number of findings in the target for this audit mitigation actions task to which mitigation actions failed when applied.                                                         |
| skippedFindingsCount         | number           | The number of findings in the target for this audit mitigation actions task to which mitigation actions were not applied because they were excluded by a filter applied to the task. |
| succeededFindingsCount       | number           | The number of findings in the target for this audit mitigation actions task to which mitigation actions were applied without error or cancellation.                                  |
| totalFindingsCount           | number           | The total number of findings in the target for this audit mitigation actions task.                                                                                                   |
| taskStatus                   | string           | The current state of the audit mitigation actions task. The status is failed if one or more of the actions for the finding failed to be applied.                                     |

## Errors

`ResourceNotFoundException`

No audit mitigation task was found with the specified `taskId`.

`InvalidRequestException`

The contents of the request were invalid.

`ThrottlingException`

The rate exceeds the limit.

`InternalFailureException`

An unexpected error has occurred.

# Detect

AWS IoT Device Defender Detect lets you identify unusual behavior that might indicate a compromised device by monitoring the behavior of your devices. Using a combination of cloud-side metrics (from AWS IoT) and device-side metrics (from agents that you install on your devices) you can detect:

- Changes in connection patterns.
- Devices that communicate to unauthorized or unrecognized endpoints.
- Changes in inbound and outbound device traffic patterns.

You create security profiles, which contain definitions of expected device behaviors, and assign them to a group of devices or to all the devices in your fleet. AWS IoT Device Defender Detect uses these security profiles to detect anomalies and send alerts through Amazon CloudWatch metrics and Amazon Simple Notification Service notifications.

AWS IoT Device Defender Detect can detect security issues frequently found in connected devices:

- Traffic from a device to a known malicious IP address or to an unauthorized endpoint that indicates a potential malicious command and control channel.
- Anomalous traffic, such as a spike in outbound traffic, that indicates a device is participating in a DDoS.
- Devices with remote management interfaces and ports that are remotely accessible.
- A spike in the rate of messages sent to your account (for example, from a rogue device that can result in excessive per-message charges).

## Use cases:

Measure attack surface

You can use AWS IoT Device Defender Detect to measure the attack surface of your devices. For example, you can identify devices with service ports that are often the target of attack campaigns (telnet service running on ports 23/2323, SSH service running on port 22, HTTP/S services running on ports 80/443/8080/8081). While these service ports might have legitimate reasons to be used on the devices, they are also usually part of the attack surface for adversaries and carry associated risks. After AWS IoT Device Defender Detect alerts you to the attack surface, you can minimize it (by eliminating unused network services) or run additional assessments to identify security weaknesses (for example, telnet configured with common, default, or weak passwords).

### Detect device behavioral anomalies with possible security root causes

You can use AWS IoT Device Defender Detect to alert you to unexpected device behavioral metrics (the number of open ports, number of connections, an unexpected open port, connections to unexpected IP addresses) that might indicate a security breach. For example, a higher than expected number of TCP connections might indicate a device is being used for a DDoS attack. A process listening on a port other than the one you expect might indicate a backdoor installed on a device for remote control. You can use AWS IoT Device Defender Detect to probe the health of your device fleets and verify your security assumptions (for example, no device is listening on port 23 or 2323).

You can enable machine learning (ML)-based threat detection to automatically identify potential threats.

### Detect an incorrectly configured device

A spike in the number or size of messages sent from a device to your account might indicate an incorrectly configured device. Such a device might increase your per-message charges. Similarly, a device with many authorization failures might require a reconfigured policy.

## Concepts

### **metric**

AWS IoT Device Defender Detect uses metrics to detect anomalous behavior. AWS IoT Device Defender Detect compares the reported value of a metric with the expected value you provide. These metrics can be taken from two sources: cloud-side metrics and device-side metrics.

Abnormal behavior on the AWS IoT network is detected by using cloud-side metrics such as the number of authorization failures, or the number or size of messages a device sends or receives through AWS IoT.

AWS IoT Device Defender Detect can also collect, aggregate, and monitor metrics data generated by AWS IoT devices (for example, the ports a device is listening on, the number of bytes or packets sent, or the device's TCP connections).

You can use AWS IoT Device Defender Detect with cloud-side metrics alone. To use device-side metrics, you must first deploy the AWS IoT SDK on your AWS IoT connected devices or device gateways to collect the metrics and send them to AWS IoT. See [Sending Metrics from Devices \(p. 656\)](#).

### **dimension**

You can define a dimension to adjust the scope of a behavior. For example, you can define a topic filter dimension that applies a behavior to MQTT topics that match a pattern. For information about defining a dimension for use in a security profile, see [CreateDimension \(p. 663\)](#).

### **security profile**

A security profile defines anomalous behaviors for a group of devices (a [thing group](#)) or for all devices in your account, and specifies which actions to take when an anomaly is detected. You can use the AWS IoT console or API commands to create a security profile and associate it with a group of devices. AWS IoT Device Defender Detect starts recording security-related data and uses the behaviors defined in the security profile to detect anomalies in the behavior of the devices.

### **behavior**

A behavior tells AWS IoT Device Defender Detect how to recognize when a device is doing something abnormal. Each behavior consists of a name, a metric, an operator, and a value or a statistical threshold. For some metrics, a time period (`durationSeconds`) is also required. Any device action that doesn't match a defined behavior statement triggers an alert.

### alert

When an anomaly is detected, an alert notification can be sent through a CloudWatch metric (see [AWS IoT Metrics \(p. 199\)](#)) or an SNS notification. An alert notification is also displayed in the AWS IoT console along with information about the alert, and a history of alerts for the device. An alert is also sent when a monitored device stops exhibiting anomalous behavior or when it had been causing an alert but stops reporting for an extended period.

## Behaviors

A security profile contains a set of behaviors. Each behavior contains a metric that specifies the normal behavior for a group of devices or for all devices in your account. See [Metrics \(p. 638\)](#) and [CreateSecurityProfile \(p. 664\)](#).

The following describes some of the fields that are used in the definition of a behavior:

#### **name**

The name for the behavior.

#### **metric**

The name of the metric used (that is, what is measured by the behavior).

#### **dimension**

You can define a dimension to adjust the scope of a behavior. For example, you can define a topic filter dimension that applies a behavior to MQTT topics that match a pattern. To define a dimension for use in a security profile, see [CreateDimension \(p. 663\)](#).

#### **criteria**

The criteria that determine if a device is behaving normally in regard to the `metric`.

#### **comparisonOperator**

The operator that relates the thing measured (`metric`) to the criteria (`value` or `statisticalThreshold`).

Possible values are: "less-than", "less-than-equals", "greater-than", "greater-than-equals", "in-cidr-set", "not-in-cidr-set", "in-port-set", and "not-in-port-set". Not all operators are valid for every metric. Operators for CIDR sets and ports are only for use with metrics involving such entities.

#### **value**

The value to be compared with the `metric`. Depending on the type of metric, this should contain a `count` (a `value`), `cids` (a list of CIDRs), or `ports` (a list of ports).

#### **statisticalThreshold**

The statistical threshold by which a behavior violation is determined. This field contains a `statistic` field that has the following possible values: "p0", "p0.1", "p0.01", "p1", "p10", "p50", "p90", "p99", "p99.9", "p99.99", or "p100".

This `statistic` indicates a percentile. It resolves to a value by which compliance with the behavior is determined. Metrics are collected one or more times over the specified duration (`durationSeconds`) from all reporting devices associated with this security profile, and percentiles are calculated based on that data. After that, measurements are collected for a device and accumulated over the same duration. If the resulting value for the device falls above

or below (`comparisonOperator`) the value associated with the percentile specified, then the device is considered to be in compliance with the behavior. Otherwise, the device is in violation of the behavior.

A `percentile` indicates the percentage of all the measurements considered that fall below the associated value. For example, if the value associated with "p90" (the 90th percentile) is 123, then 90% of all measurements were below 123.

#### **durationSeconds**

Use this to specify the period of time over which the behavior is evaluated, for those criteria that have a time dimension (for example, `NUM_MESSAGES_SENT`). For a `statisticalThreshold` metric comparison, this is the time period during which measurements are collected for all devices to determine the `statisticalThreshold` values, and then for each device to determine how its behavior ranks in comparison.

#### **consecutiveDatapointsToAlarm**

If a device is in violation of the behavior for the specified number of consecutive data points, an alarm occurs. If not specified, the default is 1. (This differs from the AWS IoT console where a value of 3 is presented by default, but can be overridden.)

#### **consecutiveDatapointsToClear**

If an alert has occurred and the offending device is no longer in violation of the behavior for the specified number of consecutive data points, the alarm is cleared. If not specified, the default is 1. (This differs from the AWS IoT console where a value of 3 is presented by default, but can be overridden.)

## Metrics

### aws:message-byte-size

The number of bytes in a message.

More info (1)

Use this metric to specify the maximum or minimum size (in bytes) of each message transmitted from a device to AWS IoT.

Source: cloud-side

Operators: less-than | less-than-equals | greater-than | greater-than-equals

Value: a non-negative integer

Units: bytes

#### Example

```
{  
  "name": "Max Message Size",  
  "metric": "aws:message-byte-size",  
  "criteria": {  
    "comparisonOperator": "less-than",  
    "value": {  
      "count": 1024  
    },  
    "consecutiveDatapointsToAlarm": 3,  
    "consecutiveDatapointsToClear": 3  
  }  
}
```

```
    }  
}
```

### Example Example using a statisticalThreshold

```
{  
  "name": "Large Message Size",  
  "metric": "aws:message-byte-size",  
  "criteria": {  
    "comparisonOperator": "less-than",  
    "statisticalThreshold": {  
      "statistic": "p90"  
    },  
    "durationSeconds": 300,  
    "consecutiveDatapointsToAlarm": 3,  
    "consecutiveDatapointsToClear": 3  
  }  
}
```

An alarm occurs for a device if, during three consecutive five-minute periods, it transmits messages whose cumulative size is more than that measured for 90 percent of all other devices reporting for this security profile behavior.

[aws:num-messages-received/aws:num-messages-sent](#)

The number of messages received or sent by a device during a given time period.

[More info \(2\)](#)

Use this metric to specify the maximum or minimum number of messages that can be sent or received between AWS IoT and each device in a given period of time.

Source: cloud-side

Operators: less-than | less-than-equals | greater-than | greater-than-equals

Value: a non-negative integer

Units: messages

Duration: a non-negative integer, valid values are 300, 600, 900, 1800 or 3600 seconds

### Example

```
{  
  "name": "Out bound message count",  
  "metric": "aws:num-messages-sent",  
  "criteria": {  
    "comparisonOperator": "less-than",  
    "value": {  
      "count": 50  
    },  
    "durationSeconds": 300,  
    "consecutiveDatapointsToAlarm": 2,  
    "consecutiveDatapointsToClear": 2  
  }  
}
```

## Example Example using a statisticalThreshold

```
{  
  "name": "Out bound message rate",  
  "metric": "aws:num-messages-sent",  
  "criteria": {  
    "comparisonOperator": "less-than",  
    "statisticalThreshold": {  
      "statistic": "p99"  
    },  
    "durationSeconds": 300,  
    "consecutiveDatapointsToAlarm": 2,  
    "consecutiveDatapointsToClear": 2  
  }  
}
```

### aws:all-bytes-out

The number of outbound bytes from a device during a given time period.

[More info \(3\)](#)

Use this metric to specify the maximum or minimum amount of outbound traffic that a device should send, measured in bytes in a given period of time.

Source: device-side

Operators: less-than | less-than-equals | greater-than | greater-than-equals

Value: a non-negative integer

Units: bytes

Duration: a non-negative integer, valid values are 300, 600, 900, 1800 or 3600 seconds

## Example

```
{  
  "name": "TCP outbound traffic",  
  "metric": "aws:all-bytes-out",  
  "criteria": {  
    "comparisonOperator": "less-than",  
    "value": {  
      "count": 4096  
    },  
    "durationSeconds": 300,  
    "consecutiveDatapointsToAlarm": 5,  
    "consecutiveDatapointsToClear": 4  
  }  
}
```

## Example Example using a statisticalThreshold

```
{  
  "name": "TCP outbound traffic",  
  "metric": "aws:all-bytes-out",  
  "criteria": {  
    "comparisonOperator": "less-than",  
  }
```

```
        "statisticalThreshold": {
            "statistic": "p50"
        },
        "durationSeconds": 900,
        "consecutiveDatapointsToAlarm": 5,
        "consecutiveDatapointsToClear": 4
    }
}
```

#### aws:all-bytes-in

The number of inbound bytes to a device during a given time period.

[More info \(4\)](#)

Use this metric to specify the maximum or minimum amount of inbound traffic that a device should receive, measured in bytes in a given period of time.

Source: device-side

Operators: less-than | less-than-equals | greater-than | greater-than-equals

Value: a non-negative integer

Units: bytes

Duration: a non-negative integer, valid values are 300, 600, 900, 1800 or 3600 seconds

#### Example

```
{
    "name": "TCP inbound traffic",
    "metric": "aws:all-bytes-in",
    "criteria": {
        "comparisonOperator": "less-than",
        "value": {
            "count": 4096
        },
        "durationSeconds": 300,
        "consecutiveDatapointsToAlarm": 1,
        "consecutiveDatapointsToClear": 3
    }
}
```

#### Example Example using a statisticalThreshold

```
{
    "name": "TCP inbound traffic",
    "metric": "aws:all-bytes-in",
    "criteria": {
        "comparisonOperator": "less-than",
        "statisticalThreshold": {
            "statistic": "p90"
        },
        "durationSeconds": 300,
        "consecutiveDatapointsToAlarm": 1,
        "consecutiveDatapointsToClear": 3
    }
}
```

aws:all-packets-out

The number of outbound packets from a device during a given time period.

[More info \(5\)](#)

Use this metric to specify the maximum or minimum amount of total outbound traffic that a device should send in a given period of time.

Source: device-side

Operators: less-than | less-than-equals | greater-than | greater-than-equals

Value: a non-negative integer

Units: packets

Duration: a non-negative integer. Valid values are 300, 600, 900, 1800 or 3600 seconds.

### Example

```
{  
  "name": "TCP outbound traffic",  
  "metric": "aws:all-packets-out",  
  "criteria": {  
    "comparisonOperator": "less-than",  
    "value": {  
      "count": 100  
    },  
    "durationSeconds": 300,  
    "consecutiveDatapointsToAlarm": 1,  
    "consecutiveDatapointsToClear": 3  
  }  
}
```

### Example Example using a statisticalThreshold

```
{  
  "name": "TCP outbound traffic",  
  "metric": "aws:all-packets-out",  
  "criteria": {  
    "comparisonOperator": "less-than",  
    "statisticalThreshold": {  
      "statistic": "p90"  
    },  
    "durationSeconds": 300,  
    "consecutiveDatapointsToAlarm": 1,  
    "consecutiveDatapointsToClear": 3  
  }  
}
```

:

aws:all-packets-in

The number of inbound packets to a device during a given time period.

[More info \(6\)](#)

Use this metric to specify the maximum or minimum amount of total inbound traffic that a device should receive in a given period of time.

Source: device-side

Operators: less-than | less-than-equals | greater-than | greater-than-equals

Value: a non-negative integer

Units: packets

Duration: a non-negative integer. Valid values are 300, 600, 900, 1800 or 3600 seconds.

**Example**

```
{  
  "name": "TCP inbound traffic",  
  "metric": "aws:all-packets-in",  
  "criteria": {  
    "comparisonOperator": "less-than",  
    "value": {  
      "count": 100  
    },  
    "durationSeconds": 300,  
    "consecutiveDatapointsToAlarm": 2,  
    "consecutiveDatapointsToClear": 1  
  }  
}
```

**Example**

Example using a statisticalThreshold

```
{  
  "name": "TCP inbound traffic",  
  "metric": "aws:all-packets-in",  
  "criteria": {  
    "comparisonOperator": "less-than",  
    "statisticalThreshold": {  
      "statistic": "p90"  
    },  
    "durationSeconds": 300,  
    "consecutiveDatapointsToAlarm": 2,  
    "consecutiveDatapointsToClear": 1  
  }  
}
```

:

[aws:num-authorization-failures](#)

The number of authorization failures during a given time period.

[More info \(7\)](#)

Use this metric to specify the maximum number of authorization failures allowed for each device in a given period of time. An authorization failure occurs when a request from a device to AWS IoT is

denied (for example, if a device attempts to publish to a topic for which it does not have sufficient permissions).

Source: cloud-side

Unit: failures

Operators: less-than | less-than-equals | greater-than | greater-than-equals

Value: a non-negative integer

Units: failures

Duration: a non-negative integer. Valid values are 300, 600, 900, 1800, or 3600 seconds.

### Example

```
{  
  "name": "Authorization Failures",  
  "metric": "aws:num-authorization-failures",  
  "criteria": {  
    "comparisonOperator": "less-than",  
    "value": {  
      "count": 5  
    },  
    "durationSeconds": 300,  
    "consecutiveDatapointsToAlarm": 2,  
    "consecutiveDatapointsToClear": 1  
  }  
}
```

### Example Example using a statisticalThreshold

```
{  
  "name": "Authorization Failures",  
  "metric": "aws:num-authorization-failures",  
  "criteria": {  
    "comparisonOperator": "less-than",  
    "statisticalThreshold": {  
      "statistic": "p50"  
    },  
    "durationSeconds": 300,  
    "consecutiveDatapointsToAlarm": 2,  
    "consecutiveDatapointsToClear": 1  
  }  
}
```

:

**aws:source-ip-address**

The IP address from which a device has connected to AWS IoT.

[More info \(8\)](#)

Use this metric to specify a set of allowed (formerly referred to as whitelisted) or denied (formerly referred to as blacklisted) CIDRs from which each device must or must not connect to AWS IoT.

Source: cloud-side

Operators: in-cidr-set | not-in-cidr-set

Values: a list of CIDRs

Units: n/a

### Example

```
{  
  "name": "Denied source IPs",  
  "metric": "aws:source-ip-address",  
  "criteria": {  
    "comparisonOperator": "not-in-cidr-set",  
    "value": {  
      "cidrs": [ "12.8.0.0/16", "15.102.16.0/24" ]  
    }  
  }  
}
```

aws:destination-ip-addresses

A set of IP destinations.

[More info \(9\)](#)

Use this metric to specify a set of allowed (formerly referred to as whitelisted) or denied (formerly referred to as blacklisted) CIDRs with which each device must or must not communicate.

Source: device-side

Operators: in-cidr-set | not-in-cidr-set

Values: a list of CIDRs

Units: n/a

Example:

### Example

```
{  
  "name": "Denied destination IPs",  
  "metric": "aws:destination-ip-addresses",  
  "criteria": {  
    "comparisonOperator": "not-in-cidr-set",  
    "value": {  
      "cidrs": [ "12.8.0.0/16", "15.102.16.0/24" ]  
    }  
  }  
}
```

aws:listening-tcp-ports / aws:listening-udp-ports

The TCP or UDP ports that the device is listening on.

[More info \(10\)](#)

Use this metric to specify a set of allowed (formerly referred to as whitelisted) or denied (formerly referred to as blacklisted) TCP/UDP ports that each device must or must not listen on.

Source: device-side

Operators: in-port-set | not-in-port-set

Values: a list of ports

Units: n/a

**Example:**

```
{  
  "name": "Listening TCP Ports",  
  "metric": "aws:listening-tcp-ports",  
  "criteria": {  
    "comparisonOperator": "in-port-set",  
    "value": {  
      "ports": [ 443, 80 ]  
    }  
  }  
}
```

[aws:num-listening-tcp-ports / aws:num-listening-udp-ports](#)

The number of TCP or UDP ports the device is listening on.

[More info \(11\)](#)

Use this metric to specify the maximum or minimum number of TCP or UDP ports that each device should listen on.

Source: device-side

Operators: less-than | less-than-equals | greater-than | greater-than-equals

Value: a non-negative integer

Units: ports

**Example**

**Example:**

**Example Example using a statisticalThreshold**

```
{  
  "name": "Max TCP Ports",  
  "metric": "aws:num-listening-tcp-ports",  
  "criteria": {  
    "comparisonOperator": "less-than-equals",  
    "statisticalThreshold": {  
      "statistic": "p90"  
    },  
    "durationSeconds": 300,  
    "consecutiveDatapointsToAlarm": 2,  
    "consecutiveDatapointsToClear": 1  
  }  
}
```

```
    }  
}
```

:

#### aws:num-established-tcp-connections

The number of TCP connections for a device.

[More info \(12\)](#)

Use this metric to specify the maximum or minimum number of active TCP connections that each device should have. (All TCP states)

Source: device-side

Operators: less-than | less-than-equals | greater-than | greater-than-equals

Value: a non-negative integer

Units: connections

#### Example

```
{  
  "name": "TCP Connection Count",  
  "metric": "aws:num-established-tcp-connections",  
  "criteria": {  
    "comparisonOperator": "less-than",  
    "value": {  
      "count": 3  
    },  
    "consecutiveDatapointsToAlarm": 3,  
    "consecutiveDatapointsToClear": 3  
  }  
}
```

#### Example Example using a statisticalThreshold

```
{  
  "name": "TCP Connection Count",  
  "metric": "aws:num-established-tcp-connections",  
  "criteria": {  
    "comparisonOperator": "less-than",  
    "statisticalThreshold": {  
      "statistic": "p90"  
    },  
    "durationSeconds": 900,  
    "consecutiveDatapointsToAlarm": 3,  
    "consecutiveDatapointsToClear": 3  
  }  
}
```

#### aws:num-connection-attempts

The number of times a device attempts to make a connection in a given time period.

[More info \(13\)](#)

Use this metric to specify the maximum or minimum number of connection attempts for each device. Successful and unsuccessful attempts are counted.

Source: cloud-side

Operators: less-than | less-than-equals | greater-than | greater-than-equals

Value: a non-negative integer

Units: connection attempts

Duration: a non-negative integer. Valid values are 300, 600, 900, 1800, or 3600 seconds.

### Example

```
{  
  "name": "Connection Attempts",  
  "metric": "aws:num-connection-attempts",  
  "criteria": {  
    "comparisonOperator": "greater-than",  
    "value": {  
      "count": 5  
    },  
    "durationSeconds": 600,  
    "consecutiveDatapointsToAlarm": 1,  
    "consecutiveDatapointsToClear": 2  
  }  
}
```

### Example Example using a statisticalThreshold

```
{  
  "name": "Connection Attempts",  
  "metric": "aws:num-connection-attempts",  
  "criteria": {  
    "comparisonOperator": "greater-than",  
    "statisticalThreshold": {  
      "statistic": "p10"  
    },  
    "durationSeconds": 300,  
    "consecutiveDatapointsToAlarm": 1,  
    "consecutiveDatapointsToClear": 2  
  }  
}
```

[aws:num-disconnects](#)

The number of times a device disconnects from AWS IoT during a given time period.

[More info \(14\)](#)

Use this metric to specify the maximum or minimum number of times a device disconnected from AWS IoT during a given time period.

Source: cloud-side

Operators: less-than | less-than-equals | greater-than | greater-than-equals

**Value:** a non-negative integer

**Units:** disconnects

**Duration:** a non-negative integer. Valid values are 300, 600, 900, 1800, or 3600 seconds.

### Example

```
{
  "name": "Disconnections",
  "metric": "aws:num-disconnects",
  "criteria": {
    "comparisonOperator": "greater-than",
    "value": {
      "count": 5
    },
    "durationSeconds": 600,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 2
  }
}
```

### Example Example using a statisticalThreshold

```
{
  "name": "Disconnections",
  "metric": "aws:num-disconnects",
  "criteria": {
    "comparisonOperator": "greater-than",
    "statisticalThreshold": {
      "statistic": "p10"
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 2
  }
}
```

## Scoping Metrics in Security Profiles Using Dimensions

Dimensions are attributes that you can define to get more precise data about metrics and behaviors in your security profile. You define the scope by providing a value or pattern that is used as a filter. For example, you can define a topic filter dimension that applies a behavior only to MQTT topics that match a particular value, such as "data/bulb/+/activity". For information about defining a dimension that you can use in your security profile, see [CreateDimension \(p. 663\)](#).

Dimension values support MQTT wildcards. MQTT wildcards help you subscribe to multiple topics simultaneously. There are two different kinds of wildcards: single-level (+) and multi-level (#). For example, the dimension value Data/bulb/+/activity creates a subscription that matches all topics that exist on the same level as the +. Dimension values also supports the MQTT client ID substitution variable \${iot:ClientId}.

Dimensions of type TOPIC\_FILTER are compatible with the following set of cloud-side metrics:

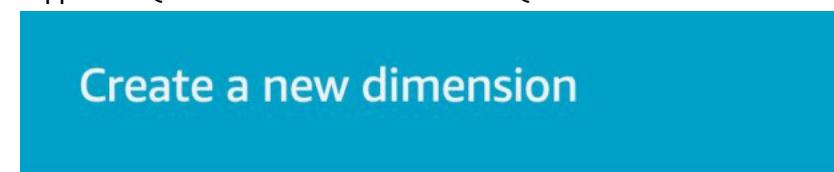
- Number of messages sent
- Number of messages received
- Message byte size
- Source IP address

- Number of authorization failures

## How to Use Dimensions in the Console

### To create and apply a dimension to a security profile behavior

1. In the [AWS IoT console](#), in the navigation pane, expand **Defend**, expand **Detect**, and select **Security profiles**.
2. On the **Security profiles** page, choose **Create** to add a new security profile, or **Edit** to apply a dimension to an existing security profile.
3. On the **Expected behaviors** page, select one of the five cloud-side metrics dimensions supports under **Metric**. The **Dimension** and **Dimension operator** boxes display. For information about supported cloud-side metrics, see [Scoping Metrics in Security Profiles Using Dimensions \(p. 649\)](#).
4. Under the **Dimension** dropdown, select **Add dimension**.
5. On the **Create a new dimension** page, enter details for your new dimension. **Dimensions values** supports MQTT wildcards # and + and the MQTT client ID substitution variable \${iot:ClientId}.

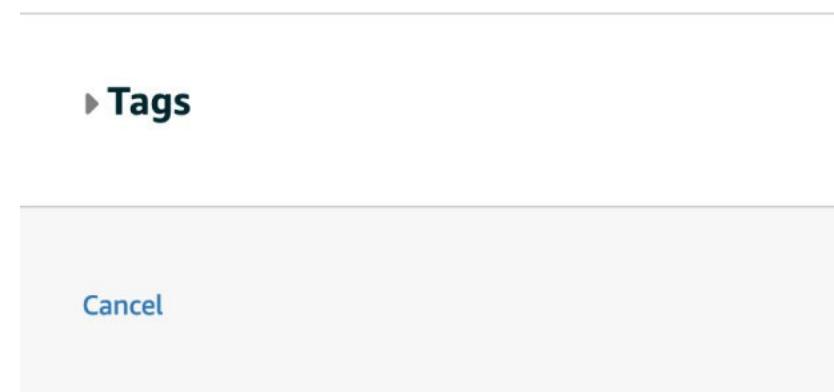


Dimension name [?](#)

Dimension type [?](#)

Topic filter

Value [?](#)



### To view your violations

1. In the [AWS IoT console](#), in the navigation pane, expand **Defend** then **Detect**. Then select **Violations**.

Device Defender > Detect > Violations

## Violations

Now History

2 Thing(s) in alarm as of Mar 27, 2020 9:24:25 AM -0700

| Event time                     | Thing name                 | Security profile |
|--------------------------------|----------------------------|------------------|
| Mar 26, 2020 10:55:00 PM -0700 | iotconsole-1585288160280-0 | test_SP          |
| Mar 26, 2020 10:55:00 PM -0700 | iotconsole-1585288160280-0 | test_SP          |

2. In the **Behavior** column, hover over the behavior you want to see the violation information for.

### To view and update your dimensions

1. In the [AWS IoT console](#), in the navigation pane, expand **Defend**, expand **Detect**, and select **Dimensions**.
2. Select the dimension that you'd like to edit.
3. Select **Actions**, **Edit**.

| Dimensions (1)                |                    |              |                       |
|-------------------------------|--------------------|--------------|-----------------------|
| Created date                  | Dimension name     | Type         | Value                 |
| Mar 27, 2020 3:22:51 PM -0700 | Sensor_Temperature | Topic filter | /sensor/temperature/+ |

### To delete a dimension

1. In the [AWS IoT console](#), in the navigation pane, expand **Defend**, expand **Detect**, and select **Dimensions**.
2. Select the dimension that you want to delete.
3. Confirm that the dimension isn't attached to a security profile by checking the **Used in** column. If the dimension is attached to a security profile, open the **Security profiles** page on the left, and edit the security profiles that the dimension is attached to. When you delete the dimension, you also delete the behavior. If you want to keep the behavior, choose the ellipsis, then choose **Copy**. Then you can proceed with deleting the behavior. If you want to delete another dimension, follow the steps in this section.

| Name                | Description (optional)        |
|---------------------|-------------------------------|
| Temperature_Profile | An optional short description |

**Behaviors**

Specify how your device **should behave**. You can use [cloud-side metrics](#) without a device agent deployed. Note: once created, behavior names cannot be edited. [?](#)

| Name <a href="#">?</a> | Metric <a href="#">?</a>          | Check Type <a href="#">?</a>                 | Operator <a href="#">?</a>                   | ... |
|------------------------|-----------------------------------|----------------------------------------------|----------------------------------------------|-----|
| Sensor_failures        | Authorization failures            | Absolute-Value                               | Greater-than                                 |     |
| <b>Value</b>           | <b>Duration <a href="#">?</a></b> | <b>Datapoints to Alarm <a href="#">?</a></b> | <b>Datapoints to Clear <a href="#">?</a></b> |     |
| 5                      | 5-minutes                         | 1                                            | 1                                            |     |

| Name <a href="#">?</a>                       | Metric <a href="#">?</a> | Dimension (optional) <a href="#">?</a> | Check Type <a href="#">?</a>                 | ... |
|----------------------------------------------|--------------------------|----------------------------------------|----------------------------------------------|-----|
| Sensor_failures                              | Authorization failures   | Select                                 | Absolute Value                               |     |
| <b>Operator <a href="#">?</a></b>            | <b>Value</b>             | <b>Duration <a href="#">?</a></b>      | <b>Datapoints to Alarm <a href="#">?</a></b> |     |
| Greater than                                 | 5                        | 5 minutes                              | 1                                            |     |
| <b>Datapoints to Clear <a href="#">?</a></b> |                          |                                        |                                              |     |
| 1                                            |                          |                                        |                                              |     |
| <b>Add behavior</b>                          |                          |                                        |                                              |     |

4. Select Actions, Delete.

## Monitoring the Behavior of Unregistered Devices

AWS IoT Device Defender Detect makes it possible to identify unusual behaviors for devices that are not registered in the AWS IoT registry. You can define security profiles that are specific to one of the following target types:

- All devices
- All registered devices (things in the AWS IoT registry)
- All unregistered devices
- Devices in a thing group

A security profile defines a set of expected behaviors for devices in your account and specifies the actions to take when an anomaly is detected. Security profiles should be attached to the most specific targets to give you granular control over which devices are being evaluated against that profile.

Unregistered devices must provide a consistent MQTT client identifier or thing name (for devices that report device metrics) over the device lifetime so all violations and metrics are attributed to the same device.

**Important**

Messages reported by devices are rejected if the thing name contains control characters or if the thing name is longer than 128 bytes of UTF-8 encoded characters.

## How to Use AWS IoT Device Defender Detect

1. You can use AWS IoT Device Defender Detect with just cloud-side metrics, but if you plan to use device-reported metrics, you must first deploy the AWS IoT SDK on your AWS IoT connected devices or device gateways. For more information, see [Sending Metrics from Devices \(p. 656\)](#).
2. Consider viewing the metrics that your devices generate before you define behaviors and create alarms. AWS IoT can collect metrics from your devices so you can first identify usual or unusual behavior for a group of devices, or for all devices in your account. Use [CreateSecurityProfile \(p. 664\)](#), but specify only those `additionalMetricsToRetain` that you're interested in. Don't specify behaviors at this point.

Use the AWS IoT console to look at your device metrics to see what constitutes typical behavior for your devices.

3. Create a set of behaviors for your security profile. Behaviors contain metrics that specify normal behavior for a group of devices or for all devices in your account. For more information and examples, see [Metrics \(p. 638\)](#). After you create a set of behaviors, you can validate them with [ValidateSecurityProfileBehaviors \(p. 678\)](#).
4. Use the [CreateSecurityProfile \(p. 664\)](#) action to create a security profile that includes your behaviors. You can use the `alertTargets` parameter to have alerts sent to a target (an SNS topic) when a device violates a behavior. (If you send alerts using SNS, be aware that these count against your AWS account's SNS topic quota. It's possible that a large burst of violations can exceed your SNS topic quota. You can also use CloudWatch metrics to check for violations. For more information, see [AWS IoT Metrics \(p. 199\)](#).)
5. Use the [AttachSecurityProfile \(p. 662\)](#) action to attach the security profile to a group of devices (a thing group), all registered things in your account, all unregistered things, or all devices. AWS IoT Device Defender Detect starts checking for abnormal behavior and, if any behavior violations are detected, sends alerts. You might want to attach a security profile to all unregistered things if, for example, you expect to interact with mobile devices that are not in your account's thing registry. You can define different sets of behaviors for different groups of devices to meet your needs.

To attach a security profile to a group of devices, you must specify the ARN of the thing group that contains them. A thing group ARN has the following format.

```
arn:aws:iot:region:account-id:thinggroup/thing-group-name
```

To attach a security profile to all of the registered things in an AWS account (ignoring unregistered things), you must specify an ARN with the following format.

```
arn:aws:iot:region:account-id:all/registered-things
```

To attach a security profile to all unregistered things, you must specify an ARN with the following format.

```
arn:aws:iot:region:account-id:all/unregistered-things
```

To attach a security profile to all devices, you must specify an ARN with the following format.

```
arn:aws:iot:region:account-id:all/things
```

6. You can also keep track of violations with the [ListActiveViolations \(p. 670\)](#) action, which lets you see which violations were detected for a given security profile or target device.

Use the [ListViolationEvents \(p. 674\)](#) action to see which violations were detected during a specified time period. You can filter these results by security profile or device.

7. If your devices violate the defined behaviors too often, or not often enough, you should fine-tune the behavior definitions.
8. To review the security profiles that you set up and the devices that are being monitored, use the [ListSecurityProfiles \(p. 672\)](#), [ListSecurityProfilesForTarget \(p. 673\)](#), and [ListTargetsForSecurityProfile \(p. 673\)](#) actions.

Use the [DescribeSecurityProfile \(p. 668\)](#) action to get more details about a security profile.

9. To update a security profile, use the [UpdateSecurityProfile \(p. 676\)](#) action. Use the [DetachSecurityProfile \(p. 669\)](#) action to detach a security profile from an account or target thing group. Use the [DeleteSecurityProfile \(p. 667\)](#) action to delete a security profile entirely.

## Permissions

This section contains information about how to set up the IAM roles and policies required to manage AWS IoT Device Defender Detect. For more information, see the [IAM User Guide](#).

### Give AWS IoT Device Defender Detect Permission to Publish Alerts to an SNS Topic

If you use the `alertTargets` parameter in [CreateSecurityProfile \(p. 664\)](#), you must specify an IAM role with two policies: a permissions policy and a trust policy. The permissions policy grants permission to AWS IoT Device Defender to publish notifications to your SNS topic. The trust policy grants AWS IoT Device Defender permission to assume the required role.

#### Permission Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "arn:aws:sns:region:account-id:your-topic-name"
      ]
    }
  ]
}
```

#### Trust Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      }
    }
  ]
}
```

```
        },
        "Action": "sts:AssumeRole"
    }
}
```

### Pass Role Policy

You also need an IAM permissions policy attached to the IAM user that allows the user to pass roles. See [Granting a User Permissions to Pass a Role to an AWS Service](#).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::account-id:role/Role_To_Pass"
        }
    ]
}
```

## Sending Metrics from Devices

AWS IoT Device Defender Detect can collect, aggregate, and monitor metrics data generated by AWS IoT devices to identify devices that exhibit abnormal behavior. This section shows you how to send metrics from a device to AWS IoT Device Defender.

You must securely deploy the AWS IoT SDK on your AWS IoT connected devices or device gateways to collect device-side metrics. AWS IoT Device Defender provides sample agents to use as examples when you create your own. If you can't provide device metrics, you can still get limited functionality based on cloud-side metrics.

Note the following:

- A sample device metric reporting agent is currently available in C at <https://github.com/aws-samples/aws-iot-device-defender-agent-c>. There is also a sample device metric reporting agent available in Python on GitHub at <https://github.com/aws-samples/aws-iot-device-defender-agent-sdk-python>. Specifically, see the `greengrass_defender_agent.py` file.
- To use the sample agents or create your own custom agent, you must install the AWS IoT Device SDK. To find resources for your development language, see [AWS IoT Core Resources](#).
- All agents must create a connection to AWS IoT and publish metrics to one of these reserved AWS IoT Device Defender MQTT topics:

```
$aws/things/THING_NAME/defender/metrics/json
```

or

```
$aws/things/THING_NAME/defender/metrics/cbor
```

AWS IoT Device Defender uses one of these topics to reply with the receipt status of your metrics reports:

```
$aws/things/THING_NAME/defender/metrics/json/accepted
```

```
$aws/things/THING_NAME/defender/metrics/cbor/accepted
```

```
$aws/things/THING_NAME/defender/metrics/json/rejected
```

```
$aws/things/THING_NAME/defender/metrics/cbor/rejected
```

- To report metrics, a device must be registered as a thing in AWS IoT.
- A device should, generally, send a metric report once every 5 minutes. Devices are throttled to one metric report every 5 minutes. (A device can't make more than one metric report every 5 minutes.)
- Devices must report current metrics. Historical metrics reporting isn't supported.
- You can use [Jobs \(p. 381\)](#) to set the metrics reporting interval of your device metric reporting agent instances. An example is included with the AWS IoT Device Defender Agent C samples. For more information, see the [README.md](#).

## Device Metrics Document Specification

### Overall Structure

| Long Name | Short Name | Required | Type   | Constraints | Notes                                           |
|-----------|------------|----------|--------|-------------|-------------------------------------------------|
| header    | hed        | Y        | Object |             | Complete block required for well-formed report. |
| metrics   | met        | Y        | Object |             | Complete block required for well-formed report. |

### Header Block

| Long Name | Short Name | Required | Type    | Constraints | Notes                                                                         |
|-----------|------------|----------|---------|-------------|-------------------------------------------------------------------------------|
| report_id | rid        | Y        | Integer |             | Monotonically increasing value. Epoch timestamp recommended.                  |
| version   | v          | Y        | String  | Major.Minor | Minor increments with addition of field. Major increments if metrics removed. |

### Metrics Block:

### TCP Connections

| Long Name               | Short Name | Parent Element          | Required | Type             | Constraints | Notes                             |
|-------------------------|------------|-------------------------|----------|------------------|-------------|-----------------------------------|
| tcp_connections         | ts         | metrics                 | N        | Object           |             |                                   |
| established_connections | te         | tcp_connections         | N        | List<Connection> |             | Established TCP state             |
| connections             | cs         | established_connections |          | List<Object>     |             |                                   |
| remote_addr             | rad        | connections             | Y        | Number           | ip:port     | IP can be IPv6 or IPv4            |
| local_port              | lp         | connections             | N        | Number           | >= 0        |                                   |
| local_interface         | li         | connections             | N        | String           |             | Interface name                    |
| total                   | t          | established_connections |          | Number           | >= 0        | Number of established connections |

### Listening TCP Ports

| Long Name           | Short Name | Parent Element      | Required | Type       | Constraints | Notes                                  |
|---------------------|------------|---------------------|----------|------------|-------------|----------------------------------------|
| listening_tcp_ports | pts        | metrics             | N        | Object     |             |                                        |
| ports               | pts        | listening_tcp_ports | N        | List<Port> | > 0         |                                        |
| port                | pt         | ports               | N        | Number     | > 0         | ports should be numbers greater than 0 |
| interface           | if         | ports               | N        | String     |             | Interface name                         |
| total               | t          | listening_tcp_ports |          | Number     | >= 0        |                                        |

### Listening UDP Ports

| Long Name           | Short Name | Parent Element      | Required | Type       | Constraints | Notes                                  |
|---------------------|------------|---------------------|----------|------------|-------------|----------------------------------------|
| listening_udp_ports | pts        | metrics             | N        | Object     |             |                                        |
| ports               | pts        | listening_udp_ports | N        | List<Port> | > 0         |                                        |
| port                | pt         | ports               | N        | Number     | > 0         | Ports should be numbers greater than 0 |
| interface           | if         | ports               | N        | String     |             | Interface name                         |

| Long Name | Short Name | Parent Element      | Required | Type   | Constraints | Notes |
|-----------|------------|---------------------|----------|--------|-------------|-------|
| total     | t          | listening_udp_ports |          | Number | >= 0        |       |

## Network Statistics

| Long Name     | Short Name | Parent Element | Required | Type   | Constraints           | Notes |
|---------------|------------|----------------|----------|--------|-----------------------|-------|
| network_stats | ns         | metrics        | N        | Object |                       |       |
| bytes_in      | bi         | network_stats  | N        | Number | Delta Metric,<br>>= 0 |       |
| bytes_out     | bo         | network_stats  | N        | Number | Delta Metric,<br>>= 0 |       |
| packets_in    | pi         | network_stats  | N        | Number | Delta Metric,<br>>= 0 |       |
| packets_out   | po         | network_stats  | N        | Number | Delta Metric,<br>>= 0 |       |

## Example

The following JSON structure uses long names.

```
{
    "header": {
        "report_id": 1530304554,
        "version": "1.0"
    },
    "metrics": {
        "listening_tcp_ports": {
            "ports": [
                {
                    "interface": "eth0",
                    "port": 24800
                },
                {
                    "interface": "eth0",
                    "port": 22
                },
                {
                    "interface": "eth0",
                    "port": 53
                }
            ],
            "total": 3
        },
        "listening_udp_ports": {
            "ports": [
                {
                    "interface": "eth0",
                    "port": 5353
                },
                {
                    "interface": "eth0",
                    "port": 67
                }
            ]
        }
    }
}
```

```
        }
    ],
    "total": 2
},
"network_stats": {
    "bytes_in": 29358693495,
    "bytes_out": 26485035,
    "packets_in": 10013573555,
    "packets_out": 11382615
},
"tcp_connections": {
    "established_connections": {
        "connections": [
            {
                "local_interface": "eth0",
                "local_port": 80,
                "remote_addr": "192.168.0.1:8000"
            },
            {
                "local_interface": "eth0",
                "local_port": 80,
                "remote_addr": "192.168.0.1:8000"
            }
        ],
        "total": 2
    }
}
}
```

### **Example JSON structure using short names**

```
{
    "hed": {
        "rid": 1530305228,
        "v": "1.0"
    },
    "met": {
        "tp": {
            "pts": [
                {
                    "if": "eth0",
                    "pt": 24800
                },
                {
                    "if": "eth0",
                    "pt": 22
                },
                {
                    "if": "eth0",
                    "pt": 53
                }
            ],
            "t": 3
        },
        "up": {
            "pts": [
                {
                    "if": "eth0",
                    "pt": 5353
                },
                {
                    "if": "eth0",
                    "pt": 67
                }
            ]
        }
    }
}
```

```
        }
    ],
    "t": 2
},
"ns": {
    "bi": 29359307173,
    "bo": 26490711,
    "pi": 10014614051,
    "po": 11387620
},
"tc": {
    "ec": {
        "cs": [
            {
                "li": "eth0",
                "lp": 80,
                "rad": "192.168.0.1:8000"
            },
            {
                "li": "eth0",
                "lp": 80,
                "rad": "192.168.0.1:8000"
            }
        ],
        "t": 2
    }
}
}
```

# Detect Commands

You can use the AWS IoT Device Defender Detect commands in this section to identify unusual behavior that might indicate a compromised device.

## AWS IoT Device Defender Detect Commands

| Name                                           | Description                                                                                                                                                                                                                                                                                          |
|------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">AttachSecurityProfile (p. 662)</a> | Associates an AWS IoT Device Defender security profile with one of the following target types: <ul style="list-style-type: none"><li>• All devices</li><li>• All registered devices (things in the AWS IoT registry)</li><li>• All unregistered devices</li><li>• Devices in a thing group</li></ul> |
| <a href="#">CreateDimension (p. 663)</a>       | Creates a dimension that you can use to limit the scope for behaviors that you define in a AWS IoT Device Defender security profile.                                                                                                                                                                 |
| <a href="#">CreateSecurityProfile (p. 664)</a> | Creates an AWS IoT Device Defender security profile.                                                                                                                                                                                                                                                 |
| <a href="#">DeleteDimension (p. 666)</a>       | Deletes a dimension if it is not referenced by a behavior in any AWS IoT Device Defender security profile.                                                                                                                                                                                           |

| Name                                                      | Description                                                                                                                          |
|-----------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">DeleteSecurityProfile (p. 667)</a>            | Deletes an AWS IoT Device Defender security profile.                                                                                 |
| <a href="#">DescribeDimension (p. 667)</a>                | Gets information about a dimension that can be used to limit the scope for behaviors in an AWS IoT Device Defender security profile. |
| <a href="#">DescribeSecurityProfile (p. 668)</a>          | Gets information about an AWS IoT Device Defender security profile.                                                                  |
| <a href="#">DetachSecurityProfile (p. 669)</a>            | Disassociates an AWS IoT Device Defender security profile from a thing group or from this AWS account.                               |
| <a href="#">ListActiveViolations (p. 670)</a>             | Lists the active violations for a given AWS IoT Device Defender security profile.                                                    |
| <a href="#">ListDimensions (p. 671)</a>                   | Lists the dimensions defined in your AWS account.                                                                                    |
| <a href="#">ListSecurityProfiles (p. 672)</a>             | Lists the AWS IoT Device Defender security profiles for your AWS account, including those profiles that reference a dimension.       |
| <a href="#">ListSecurityProfilesForTarget (p. 673)</a>    | Lists the AWS IoT Device Defender security profiles attached to a target (thing group).                                              |
| <a href="#">ListTargetsForSecurityProfile (p. 673)</a>    | Lists the targets (thing groups) associated with a given AWS IoT Device Defender security profile.                                   |
| <a href="#">ListViolationEvents (p. 674)</a>              | Lists the AWS IoT Device Defender security profile violations discovered during the given time period.                               |
| <a href="#">UpdateDimension (p. 675)</a>                  | Updates the value of a defined dimension for use in the behaviors of your security profiles.                                         |
| <a href="#">UpdateSecurityProfile (p. 676)</a>            | Updates an AWS IoT Device Defender security profile.                                                                                 |
| <a href="#">ValidateSecurityProfileBehaviors (p. 678)</a> | Validates an AWS IoT Device Defender security profile behaviors specification.                                                       |

## AttachSecurityProfile

Associates an AWS IoT Device Defender security profile with one of the following target types:

- All devices
- All registered devices (things in the AWS IoT registry)
- All unregistered devices
- Devices in a thing group

Each target type can have up to five security profiles associated with it.

For more information, see [AttachSecurityProfile](#) in the API reference.

**Synopsis:**

```
aws iot attach-security-profile \
--security-profile-name <value> \
--security-profile-target-arn <value> \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json format:

```
{  
    "securityProfileName": "string",  
    "securityProfileTargetArn": "string"  
}
```

To attach a security profile to a group of devices, you must specify the ARN of the thing group that contains them. A thing group ARN has the following format.

```
arn:aws:iot:<region>:<account-id>:thinggroup/<thing-group-name>
```

To attach a security profile to all of the registered things in an account (ignoring unregistered things), you must specify an ARN with the following format.

```
arn:aws:iot:<region>:<account-id>:all/registered-things
```

To attach a security profile to all unregistered things, you must specify an ARN with the following format.

```
arn:aws:iot:<region>:<account-id>:all/unregistered-things
```

To attach a security profile to all devices, you must specify an ARN with the following format.

```
arn:aws:iot:<region>:<accountid>:all/things
```

## CreateDimension

Create a dimension to control the scope for the behavior in a security profile for AWS IoT Device Defender. The scope is defined by providing a value or pattern that is used as a filter. TOPIC\_FILTER is the only dimension currently supported. For TOPIC\_FILTER dimensions, this filter specifies the subset of MQTT topics to which the behavior is applied. MQTT wildcards # and +, and also the \${iot:ClientId} variable are supported by dimensions.

To add a dimension to a security profile, see [CreateSecurityProfile \(p. 664\)](#).

For more information, see [CreateDimension](#) in the API reference.

**Synopsis:**

```
aws iot create-dimension \
--name <value> \
--type <value> \
--string-values <value> \
[--client-request-token <value>] \
[--expected-version <value>] \
[--cli-input-json <value>]
```

```
[--generate-cli-skeleton]
```

**Note**

If you omit the `clientRequestToken`, a unique client request token is generated automatically.

**Example**

This example creates a dimension named `TopicFilterForAuthMessages` that filters to MQTT topics that match the pattern `device/+auth`. You apply the dimension to a behavior in a security profile.

```
aws iot create-dimension \
--name TopicFilterForAuthMessages --type TOPIC_FILTER --string-values device/+auth
```

The response resembles the following.

```
{
  "name": "TopicFilterForAuthMessages",
  "arn": "arn:aws:iot:eu-west-2:123456789012:dimension/TopicFilterForAuthMessages"
}
```

**Example**

This example creates a dimension named `MyFavoriteTopics` with multiple topics.

```
aws iot create-dimension \
--name MyFavoriteTopics --type TOPIC_FILTER \ --string-values topic1 topic2
```

cli-input-json format:

```
{
  "clientRequestToken": "string",
  "tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ],
  "type": "string",
  "string-values": "string"
}
```

Output:

```
{
  "arn": "string",
  "name": "string"
}
```

## CreateSecurityProfile

Creates an AWS IoT Device Defender security profile.

For more information, see [CreateSecurityProfile](#) in the API reference.

**Synopsis:**

```
aws iot create-security-profile \
--security-profile-name <value> \
[--security-profile-description <value>] \
[--behaviors <value>] \
[--alert-targets <value>] \
[--additional-metrics-to-retain <value>] \
[--tags <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

**Example**

To define a behavior with a dimension, use the `MetricDimension` field next to the metric. To retain a metric with a dimension, use the `AdditionalMetricsToRetainV2` field. `MetricDimension` has an `operator` field that allows inverting topic filters.

The following example shows a definition that there should be no messages sent to dimensions other than `FavoriteTopics`. Also, this definition enables retaining metrics for the number of messages sent to the auth topic.

```
aws iot create-security-profile \
--security-profile-name security-profile-for-smart-lights \
--behaviors '[{
"name": "num-messages-sent-to-unexpected-topic",
"metric": "aws:num-messages-sent",
"metricDimension": {
    "dimensionName": "FavoriteTopics",
    "operator": "NOT_IN"
},
"criteria": {
    "comparisonOperator": "less-than",
    "value": {"count": 1},
    "durationSeconds": 300
}']}' \
--additional-metrics-to-retain-v2 '[{
    "metric": "aws:num-messages-sent",
    "metricDimension": {"dimensionName": "TopicFilterForAuthMessages"}
}]'
```

**Example**

`cli-input-json` format:

```
{
    "additionalMetricsToRetainV2": [ "string" ],
    "alertTargets": {
        "string" : {
            "alertTargetArn": "string",
            "roleArn": "string"
        }
    },
    "behaviors": [
        {
            "criteria": {
                "comparisonOperator": "string",
                "consecutiveDatapointsToAlarm": number,
                "consecutiveDatapointsToClear": number,
                "dimensions": [
                    {
                        "dimensionName": "string",
                        "operator": "string"
                    }
                ],
                "metric": "string",
                "metricDimension": {
                    "dimensionName": "string"
                }
            }
        }
    ]
}
```

```
"durationSeconds": number,
"statisticalThreshold": {
    "statistic": "string"
},
"value": {
    "cidrs": [ "string" ],
    "count": number,
    "ports": [ number ]
}
},
"metricDimension": [ "string" ],
"metric": "string",
"name": "string"
},
],
"securityProfileDescription": "string",
"tags": [
{
    "Key": "string",
    "Value": "string"
}
]
}
```

Output:

```
{
    "securityProfileName": "string",
    "securityProfileArn": "string"
}
```

## DeleteDimension

Deletes an AWS IoT Device Defender dimension. You can delete only the dimensions that are not being used in any security profile.

To delete a dimension, you must first detach the dimension from the behavior in the security profile. For information, see [UpdateSecurityProfile \(p. 676\)](#).

For more information, see [DeleteDimension](#) in the API reference.

### Synopsis:

```
aws iot delete-dimension \
--name <value> \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

### Example

This example deletes the dimension named TopicFilterForAuthMessages.

```
aws iot delete-dimension --name TopicFilterForAuthMessages
```

This command returns no response.

cli-input-json format:

```
{  
    "name": "string",  
    "arn": "string",  
    "type": "TOPIC_FILTER",  
    "description": "string"  
}
```

Output:

None

## DeleteSecurityProfile

Deletes an AWS IoT Device Defender security profile.

For more information, see [DeleteSecurityProfile](#) in the API reference.

**Synopsis:**

```
aws iot delete-security-profile \  
    --security-profile-name <value> \  
    [--expected-version <value>] \  
    [--cli-input-json <value>] \  
    [--generate-cli-skeleton]
```

cli-input-json format:

```
{  
    "securityProfileName": "string",  
    "expectedVersion": "long"  
}
```

Output:

None

## DescribeDimension

Gets information about a dimension that can be used within behaviors in an AWS IoT Device Defender security profile.

For more information, see [DescribeDimension](#) in the API reference.

**Synopsis:**

```
aws iot describe-dimension \  
    --name <value> \  
    [--cli-input-json <value>] \  
    [--generate-cli-skeleton]
```

### Example

This example gets the definition for the dimension named *TopicFilterForAuthMessages*.

```
aws iot describe-dimension --name TopicFilterForAuthMessages
```

The response resembles the following,

```
{  
    "name": "TopicFilterForAuthMessages",  
    "arn": "arn:aws:iot:eu-west-2:1123456789012:dimension/TopicFilterForAuthMessages",  
    "type": "TOPIC_FILTER",  
    "stringValues": [  
        "device/+auth"  
    ],  
    "creationDate": 1578620223.255,  
    "lastModifiedDate": 1578620223.255  
}
```

cli-input-json format:

```
{  
    "name": "string"  
}
```

**Output:**

```
{  
    "arn": "string",  
    "creationDate": number,  
    "lastModifiedDate": number,  
    "name": "string",  
    "type": "string",  
    "string-values": "string",  
    "additional-metrics-to-retain-v2": [ "string" ]  
}
```

## Example

```
aws iot describe-dimension --name myAdminTopicDimension
```

The response looks like the following.

```
{  
    "name": "TopicFilterForAuthMessages",  
    "arn": "arn:aws:iot:eu-west-2:123456789012:dimension/TopicFilterForAuthMessages",  
    "type": "TOPIC_FILTER",  
    "string-values": "device/+auth",  
    "creationDate": 1578620223.255,  
    "lastModifiedDate": 1578620223.255  
}
```

## DescribeSecurityProfile

Gets information about an AWS IoT Device Defender security profile.

For more information, see [DescribeSecurityProfile](#) in the API reference.

**Synopsis:**

```
aws iot describe-security-profile \  
    --security-profile-name <value> \  
    [--cli-input-json <value>] \  
    [--generate-cli-skeleton]
```

cli-input-json format:

```
{  
    "securityProfileName": "string"  
}
```

**Output:**

```
{  
    "securityProfileName": "string",  
    "securityProfileArn": "string",  
    "securityProfileDescription": "string",  
    "behaviors": [  
        {  
            "name": "string",  
            "metricDimension": [ "string "],  
            "metric": "string",  
            "criteria": {  
                "comparisonOperator": "string",  
                "value": {  
                    "count": "long",  
                    "cidrs": [  
                        "string"  
                    ],  
                    "ports": [  
                        "integer"  
                    ]  
                },  
                "durationSeconds": "integer",  
                "consecutiveDatapointsToAlarm": "integer",  
                "consecutiveDatapointsToClear": "integer",  
                "statisticalThreshold": {  
                    "statistic": "string"  
                }  
            }  
        }  
    ],  
    "alertTargets": {  
        "string": {  
            "alertTargetArn": "string",  
            "roleArn": "string"  
        }  
    },  
    "additionalMetricsToRetain": [  
        "string"  
    ],  
    "version": "long",  
    "creationDate": "timestamp",  
    "lastModifiedDate": "timestamp"  
}
```

## DetachSecurityProfile

Disassociates an AWS IoT Device Defender security profile from a thing group or from this AWS account.

For more information, see [DetachSecurityProfile](#) in the API reference.

**Synopsis:**

```
aws iot detach-security-profile \  
    --security-profile-name <value> \  
    --security-profile-target-arn <value> \  
    --region <value>
```

```
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "securityProfileName": "string",
  "securityProfileTargetArn": "string"
}
```

**Output:**

None

## ListActiveViolations

Lists the active violations for a given AWS IoT Device Defender security profile.

For more information, see [ListActiveViolations](#) in the API reference.

**Synopsis:**

```
aws iot list-active-violations \
[--thing-name <value>] \
[--security-profile-name <value>] \
[--next-token <value>] \
[--max-results <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "thingName": "string",
  "securityProfileName": "string",
  "nextToken": "string",
  "maxResults": "integer"
}
```

**Output:**

```
{
  "activeViolations": [
    {
      "violationId": "string",
      "thingName": "string",
      "securityProfileName": "string",
      "behavior": {
        "name": "string",
        "dimensionNames": [ "string" ],
        "metric": "string",
        "criteria": {
          "comparisonOperator": "string",
          "value": {
            "count": "long",
            "cidrs": [
              "string"
            ],
            "ports": [
              "integer"
            ]
          }
        }
      }
    }
  ]
}
```

```
        ],
    },
    "durationSeconds": "integer",
    "consecutiveDatapointsToAlarm": "integer",
    "consecutiveDatapointsToClear": "integer",
    "statisticalThreshold": {
        "statistic": "string"
    }
}
},
"lastViolationValue": {
    "count": "long",
    "cidrs": [
        "string"
    ],
    "ports": [
        "integer"
    ],
    "lastViolationTime": "timestamp",
    "violationStartTime": "timestamp"
}
],
"nextToken": "string"
}
```

## ListDimensions

Lists the dimensions defined in your AWS account that you can use in your AWS IoT Device Defender security profiles.

To see all security profiles where a dimension is applied, use [ListSecurityProfiles \(p. 672\)](#).

For more information, see [ListDimensions](#) in the API reference.

### Synopsis:

```
aws iot list-dimensions \
[--next-token <value>] \
[--max-results <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

### Example

This example lists all dimensions defined for your AWS account.

```
aws iot list-dimensions
```

cli-input-json format:

```
{
    "nextToken": "string",
    "maxResults": "integer"
}
```

### Output:

```
{
```

```
    "dimensionNames": [ "string" ],
    "nextToken": "string"
}
```

## ListSecurityProfiles

Lists the AWS IoT Device Defender security profiles for your AWS account. You can use filters to list only those security profiles associated with a thing group or only those associated with your account.

For more information, see [ListSecurityProfiles](#) in the API reference.

### Synopsis:

```
aws iot list-security-profiles \
[--next-token <value>] \
[--max-results <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

### Example

The following example lists all security profiles that have behaviors that use the dimension named *TopicFilterForAuthMessages*.

```
aws iot list-security-profiles \
--dimension-name TopicFilterForAuthMessages
```

The response looks like the following.

```
{
  "securityProfileIdentifiers": [
    {
      "name": "ExpectedBehaviorsForSmartDevice",
      "arn": "arn:aws:iot:eu-west-2:123456789012:securityprofile/
ExpectedBehaviorsForSmartDevice"
    }
  ]
}
```

cli-input-json format:

```
{
  "nextToken": "string",
  "maxResults": "integer"
}
```

### Output:

```
{
  "securityProfileIdentifiers": [
    {
      "name": "string",
      "arn": "string"
    },
    "nextToken": "string"
}
```

## ListSecurityProfilesForTarget

Lists the AWS IoT Device Defender security profiles attached to a target (thing group).

For more information, see [ListSecurityProfilesForTarget](#) in the API reference.

### Synopsis:

```
aws iot list-security-profiles-for-target \
[--next-token <value>] \
[--max-results <value>] \
[--recursive | --no-recursive] \
--security-profile-target-arn <value> \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json format:

```
{  
    "nextToken": "string",  
    "maxResults": "integer",  
    "recursive": "boolean",  
    "securityProfileTargetArn": "string"  
}
```

### Output:

```
{  
    "securityProfileTargetMappings": [  
        {  
            "securityProfileIdentifier": {  
                "name": "string",  
                "arn": "string"  
            },  
            "target": {  
                "arn": "string"  
            }  
        }  
    ],  
    "nextToken": "string"  
}
```

## ListTargetsForSecurityProfile

Lists the targets (thing groups) associated with a given AWS IoT Device Defender security profile.

For more information, see [ListTargetsForSecurityProfile](#) in the API reference.

### Synopsis:

```
aws iot list-targets-for-security-profile \
--security-profile-name <value> \
[--next-token <value>] \
[--max-results <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json format:

```
{  
    "securityProfileName": "string",  
    "nextToken": "string",  
    "maxResults": "integer"  
}
```

**Output:**

```
{  
    "securityProfileTargets": [  
        {  
            "arn": "string"  
        }  
    ],  
    "nextToken": "string"  
}
```

## ListViolationEvents

Lists the AWS IoT Device Defender security profile violations discovered during the given time period. You can use filters to limit the results to those alerts issued for a particular security profile, behavior or thing (device).

For more information, see [ListViolationEvents](#) in the API reference.

**Synopsis:**

```
aws iot list-violation-events \  
    --start-time <value> \  
    --end-time <value> \  
    [--thing-name <value>] \  
    [--security-profile-name <value>] \  
    [--next-token <value>] \  
    [--max-results <value>] \  
    [--cli-input-json <value>] \  
    [--generate-cli-skeleton]
```

**cli-input-json format:**

```
{  
    "startTime": "timestamp",  
    "endTime": "timestamp",  
    "thingName": "string",  
    "securityProfileName": "string",  
    "nextToken": "string",  
    "maxResults": "integer"  
}
```

**Output:**

```
{  
    "violationEvents": [  
        {  
            "violationId": "string",  
            "thingName": "string",  
            "securityProfileName": "string",  
            "behavior": {  
                "name": "string",  
                "dimensionNames": [ "string" ],  
            },  
            "time": "timestamp",  
            "severity": "string",  
            "resourceType": "string",  
            "resourceArn": "string",  
            "behaviorType": "string",  
            "dimensions": [ { "name": "string", "value": "string" } ],  
            "lastUpdated": "timestamp",  
            "status": "string",  
            "rule": "string",  
            "ruleVersion": "string",  
            "ruleARN": "string",  
            "ruleStatus": "string",  
            "ruleType": "string",  
            "ruleDimensions": [ { "name": "string", "value": "string" } ]  
        }  
    ]  
}
```

```

    "metric": "string",
    "criteria": {
        "comparisonOperator": "string",
        "value": {
            "count": "long",
            "cidrs": [
                "string"
            ],
            "ports": [
                "integer"
            ]
        },
        "durationSeconds": "integer",
        "consecutiveDatapointsToAlarm": "integer",
        "consecutiveDatapointsToClear": "integer",
        "statisticalThreshold": {
            "statistic": "string"
        }
    },
    "metricValue": {
        "count": "long",
        "cidrs": [
            "string"
        ],
        "ports": [
            "integer"
        ],
        "violationEventType": "string",
        "violationEventTime": "timestamp"
    }
},
"nextToken": "string"
}

```

## UpdateDimension

Update a dimension to change the scope for the metric in a security profile for AWS IoT Device Defender. The scope is defined by providing a value or pattern that is used as a filter. For TOPIC\_FILTER dimensions, this filter specifies the subset of MQTT topics to which the behavior is applied.

For more information, see [UpdateDimension](#) in the API reference.

### Synopsis:

```

aws iot update-dimension \
--name <value> \
--string-values <value> \
[--client-request-token <value>] \
[--expected-version <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]

```

### Example

This example updates a dimension named TopicFilterForAuthMessages, setting the pattern to that filters to MQTT topics that match the pattern device/+/auth. You apply the dimension to a behavior in a security profile.

```

aws iot update-dimension \

```

```
--name TopicFilterForAuthMessages --string-values device/+/auth
```

The response resembles the following.

```
{  
    "arn": "arn:aws:iot:eu-west-2:123456789012:dimension/TopicFilterForAuthMessages",  
    "creationDate": 1578620223.255,  
    "lastModifiedDate": 1578620223.255,  
    "name": "TopicFilterForAuthMessages",  
    "type": "TOPIC_FILTER",  
    "string-values": "device/+/auth"  
}
```

cli-input-json format:

```
{  
    "string-values": "string"  
}
```

Output:

```
{  
    "arn": "string",  
    "creationDate": number,  
    "lastModifiedDate": number,  
    "name": "string",  
    "type": "string",  
    "string-values": "string"  
}
```

## UpdateSecurityProfile

Updates an AWS IoT Device Defender security profile.

For more information, see [UpdateSecurityProfile](#) in the API reference.

**Synopsis:**

```
aws iot update-security-profile \  
    --security-profile-name <value> \  
    [--security-profile-description <value>] \  
    [--behaviors <value>] \  
    [--alert-targets <value>] \  
    [--additional-metrics-to-retain <value>] \  
    [--delete-behaviors | --no-delete-behaviors] \  
    [--delete-alert-targets | --no-delete-alert-targets] \  
    [--delete-additional-metrics-to-retain | --no-delete-additional-metrics-to-retain] \  
    [--expected-version <value>] \  
    [--cli-input-json <value>] \  
    [--generate-cli-skeleton]
```

cli-input-json format:

```
{
    "securityProfileName": "string",
    "securityProfileDescription": "string",
    "behaviors": [
        {
            "name": "string",
            "metricDimension": [ "string" ],
            "metric": "string",
            "criteria": {
                "comparisonOperator": "string",
                "value": {
                    "count": "long",
                    "cidrs": [
                        "string"
                    ],
                    "ports": [
                        "integer"
                    ]
                },
                "durationSeconds": "integer",
                "consecutiveDatapointsToAlarm": "integer",
                "consecutiveDatapointsToClear": "integer",
                "statisticalThreshold": {
                    "statistic": "string"
                }
            }
        }
    ],
    "alertTargets": {
        "string": {
            "alertTargetArn": "string",
            "roleArn": "string"
        }
    },
    "additionalMetricsToRetain": [
        "string"
    ],
    "deleteBehaviors": "boolean",
    "deleteAlertTargets": "boolean",
    "deleteAdditionalMetricsToRetain": "boolean",
    "expectedVersion": "long"
}
```

#### Output:

```
{
    "securityProfileName": "string",
    "securityProfileArn": "string",
    "securityProfileDescription": "string",
    "behaviors": [
        {
            "name": "string",
            "metricDimension": [ "string" ],
            "metric": "string",
            "criteria": {
                "comparisonOperator": "string",
                "value": {
                    "count": "long",
                    "cidrs": [
                        "string"
                    ],
                    "ports": [
                        "integer"
                    ]
                }
            }
        }
    ]
}
```

```

        },
        "durationSeconds": "integer",
        "consecutiveDatapointsToAlarm": "integer",
        "consecutiveDatapointsToClear": "integer",
        "statisticalThreshold": {
            "statistic": "string"
        }
    }
],
"alertTargets": {
    "string": {
        "alertTargetArn": "string",
        "roleArn": "string"
    }
},
"additionalMetricsToRetain": [
    "string"
],
"version": "long",
"creationDate": "timestamp",
"lastModifiedDate": "timestamp"
}
}

```

## ValidateSecurityProfileBehaviors

Validates an AWS IoT Device Defender security profile behaviors specification.

For more information, see [ValidateSecurityProfileBehaviors](#) in the API reference.

### Synopsis:

```
aws iot validate-security-profile-behaviors \
--behaviors <value> \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

**cli-input-json** format:

```
{
    "behaviors": [
        {
            "name": "string",
            "metricDimension": [ "string" ],
            "metric": "string",
            "criteria": {
                "comparisonOperator": "string",
                "value": {
                    "count": "long",
                    "cidrs": [
                        "string"
                    ],
                    "ports": [
                        "integer"
                    ]
                },
                "durationSeconds": "integer",
                "consecutiveDatapointsToAlarm": "integer",
                "consecutiveDatapointsToClear": "integer",
                "statisticalThreshold": {
                    "statistic": "string"
                }
            }
        }
    ]
}
```

```
        }
    ]
}
```

Output:

```
{
  "valid": "boolean",
  "validationErrors": [
    {
      "errorMessage": "string"
    }
  ]
}
```

## Device Agent Integration with AWS IoT Greengrass

AWS IoT Device Defender can be used with AWS IoT Greengrass. Device agent integration follows the standard AWS IoT Greengrass Lambda function deployment model, allowing you to add AWS IoT Device Defender security to your AWS IoT Greengrass core devices. Follow these steps to integrate a device agent.

Prerequisites:

- Set up your AWS IoT Greengrass environment.
- Configure and run your AWS IoT Greengrass core.
- Ensure you can successfully deploy and run a Lambda function on your AWS IoT Greengrass core.

In general, the process described here follows the [Create and Package a Lambda Function](#) section in the [AWS IoT Greengrass Developer Guide](#).

### To create a Lambda package

1. Clone the AWS IoT Device Defender Python samples repository.

```
git clone https://github.com/aws-samples/aws-iot-device-defender-agent-sdk-python.git
```

2. Create and activate a virtual environment (optional, but recommended).

```
pip install virtualenv
virtualenv metrics_lambda_environment
source metrics_lambda_environment/bin/activate
```

3. Install the AWS IoT Device Defender sample agent in the virtual environment. Install from PyPi.

```
pip install AWSIoTDeviceDefenderAgentSDK
```

4. Install the downloaded source.

```
cd aws-iot-device-defender-agent-sdk-python
#This must be run from the same directory as setup.py
pip install .
```

5. Create an empty directory to assemble your Lambda function. This is your Lambda directory.

```
mkdir metrics_lambda
cd metrics_lambda
```

6. Complete steps 1-4 in [Create and Package a Lambda Function](#).
7. Unzip the AWS IoT Greengrass Python SDK into your Lambda directory.

```
unzip ../aws_greengrass_core_sdk/sdk/python_sdk_1_1_0.zip
cp -R ../aws_greengrass_core_sdk/examples/HelloWorld/greengrass_common .
cp -R ../aws_greengrass_core_sdk/examples/HelloWorld/greengrassdk .
cp -R ../aws_greengrass_core_sdk/examples/HelloWorld/greengrass_ipc_python_sdk .
```

8. Copy the AWSIoTDeviceDefenderAgentSDK module to the root level of your Lambda directory.

```
cp -R ../aws-iot-device-defender-agent-sdk-python/AWSIoTDeviceDefenderAgentSDK .
```

9. Copy the AWS IoT Greengrass agent to the root level of your Lambda directory.

```
cp ../aws-iot-device-defender-agent-sdk-python/samples/greengrass/
greengrass_core_metrics_agent/greengrass_defender_agent.py .
```

10. Customize the AWS IoT Greengrass agent to include the name of your AWS IoT Greengrass core device and the desired metrics sample rate:

- Replace GREENGASS\_CORENAME with the name of your AWS IoT Greengrass core.
- Set the SAMPLE\_RATE\_SECONDS to your desired metrics reporting interval. The shortest reporting interval supported by AWS IoT Device Defender is 5 minutes (300 seconds).

11. Copy the dependencies from your virtual environment (or your system) into the root level of your Lambda directory.

```
cp -R ../metrics_lambda_environment/lib/python2.7/site-packages/psutil .
cp -R ../metrics_lambda_environment/lib/python2.7/site-packages/cbor .
```

12. Create your Lambda function zip file. Perform this command in the root level of your Lambda directory.

```
rm *.zip
zip -r greengrass_defender_metrics_lambda.zip *
```

## To configure and deploy your AWS IoT Greengrass Lambda function

1. [Upload your lambda zip file](#).
2. Select the Python 2.7 runtime, and in the Handler field, enter `greengrass_defender_agent.function_handler`.
3. [Configure your Lambda function as a long-lived Lambda function](#).
4. [Configure a subscription from your Lambda function to the AWS IoT cloud](#). For AWS IoT Device Defender, a subscription from the AWS Cloud to your Lambda function is not required.
5. Create a local resource to allow your Lambda function to collect metrics from your AWS IoT Greengrass core host:
  - Follow the instructions in [Access Local Resources with Lambda Functions](#). Use the following parameters:
    - Resource Name: Core\_Proc
    - Type: Volume

- Source Path: /proc
  - Destination Path: /host\_proc
  - Group owner file access permission: Automatically add OS group permissions of the Linux group that owns the resource
  - Associate the resource with your metrics Lambda function.
6. Deploy your Lambda function to your AWS IoT Greengrass group.

#### To review your AWS IoT Device Defender device metrics using the AWS IoT console

1. Temporarily modify the publish topic in your AWS IoT Greengrass Lambda function to "metrics/test".
2. Deploy the Lambda function.
3. To see the metrics your AWS IoT Greengrass core is emitting, on the **Test** page of the AWS IoT console, add a subscription to the temporary topic ("metrics/test").

## Security Best Practices for Device Agents

### Least Privilege

The agent process should be granted only the minimum permissions required to perform its duties.

### Basic Mechanisms

- Agent should be run as non-root user.
- Agent should run as a dedicated user, in its own group.
- User/groups should be granted read-only permissions on the resources required to gather and transmit metrics.
- Example: read-only on /proc /sys for the sample agent.
- For an example of how to set up a process to run with reduced permissions, see the setup instructions that are included with the Python sample agent.

There are a number of well-known Linux mechanisms that can help you further restrict or isolate your agent process:

### Advanced Mechanisms

- [CGroups](#)
- [SELinux](#)
- [Chroot](#)
- [Linux Namespaces](#)

### Operational Resiliency

An agent process must be resilient to unexpected operational errors and exceptions and must not crash or exit permanently. The code needs to gracefully handle exceptions and, as a precaution, it must be configured to automatically restart in case of unexpected termination (for example, due to system restarts or uncaught exceptions).

### Least Dependencies

An agent must use the least possible number of dependencies (that is, third-party libraries) in its implementation. If use of a library is justified due to the complexity of a task (for example, transport layer security), use only well-maintained dependencies and establish a mechanism to keep them up to date. If the added dependencies contain functionality not used by the agent and active by default (for example, opening ports, domain sockets), disable them in your code or by means of the library's configuration files.

### Process Isolation

An agent process must only contain functionality required for performing device metric gathering and transmission. It must not piggyback on other system processes as a container or implement functionality for other out of scope use cases. In addition, the agent process must refrain from creating inbound communication channels such as domain socket and network service ports that would allow local or remote processes to interfere with its operation and impact its integrity and isolation.

### Stealthiness

An agent process must not be named with keywords such as security, monitoring, or audit indicating its purpose and security value. Generic code names or random and unique-per-device process names are preferred. The same principle must be followed in naming the directory in which the agent's binaries reside and any names and values of process arguments.

### Least Information Shared

Any agent artifacts deployed to devices must not contain sensitive information such as privileged credentials, debugging and dead code, or inline comments or documentation files that reveal details about server-side processing of agent-gathered metrics or other details about backend systems.

### Transport Layer Security

To establish TLS secure channels for data transmission, an agent process must enforce all client-side validations, such as certificate chain and domain name validation, at the application level, if not enabled by default. Furthermore, an agent must use a root certificate store that contains trusted authorities and does not contain certificates belonging to compromised certificate issuers.

### Secure Deployment

Any agent deployment mechanism, such as code push or sync and repositories containing its binaries, source code and any configuration files (including trusted root certificates), must be access-controlled to prevent unauthorized code injection or tampering. If the deployment mechanism relies on network communication, then use cryptographic methods to protect the integrity of deployment artifacts in transit.

### Further Reading

- [Security and Identity for AWS IoT](#)
- [Understanding the AWS IoT Security Model](#)
- [Redhat: A Bite of Python](#)
- [10 common security gotchas in Python and how to avoid them](#)
- [What Is Least Privilege & Why Do You Need It?](#)
- [OWASP Embedded Security Top 10](#)
- [OWASP IoT Project](#)

# Event Messages

This section contains information about messages published by AWS IoT when things or jobs are updated or changed. For information about the AWS IoT Events service that allows you to create detectors to monitor your devices for failures or changes in operation, and to trigger actions when they occur, see [AWS IoT Events](#).

AWS IoT publishes event messages when certain events occur. For example, events are generated by the 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.

**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.

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.

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/*"
            ]
        }
    ]
}
```

You control which event types are published by calling the [UpdateEventConfigurations](#) API or by using the **update-event-configurations** CLI command. For example:

```
aws iot update-event-configurations --event-configurations "{\"THING\":{\"Enabled\": true}}"
```

**Note**

All quotation marks ("") are escaped with a backslash (\).

You can get the current event configuration by calling the [DescribeEventConfigurations](#) API or by using the **describe-event-configurations** CLI command. For example:

```
aws iot describe-event-configurations
```

The output of the **describe-event-configurations** command looks like the following:

```
{
    "lastModifiedDate": 1552671347.841,
    "eventConfigurations": {
        "THING_TYPE": {
            "Enabled": false
        },
        "JOB_EXECUTION": {

```

```

        "Enabled": false
    },
    "THING_GROUP_HIERARCHY": {
        "Enabled": false
    },
    "CERTIFICATE": {
        "Enabled": false
    },
    "THING_TYPE_ASSOCIATION": {
        "Enabled": false
    },
    "THING_GROUP_MEMBERSHIP": {
        "Enabled": false
    },
    "CA_CERTIFICATE": {
        "Enabled": false
    },
    "THING": {
        "Enabled": true
    },
    "JOB": {
        "Enabled": false
    },
    "POLICY": {
        "Enabled": false
    },
    "THING_GROUP": {
        "Enabled": false
    }
},
"creationDate": 1552671347.84
}

```

## Registry Events

The registry publishes event messages when things, thing types, and thing groups are created, updated, or deleted. The registry currently supports the following event types:

### Thing Created/Updated/Deleted

The registry publishes the following event messages when things are created, updated, or deleted:

- \$aws/events/thing/<thingName>/created
- \$aws/events/thing/<thingName>/updated
- \$aws/events/thing/<thingName>/deleted

The messages contain the following example payload:

```
{
    "eventType" : "THING_EVENT",
    "eventId" : "f5ae9b94-8b8e-4d8e-8c8f-b3266dd89853",
    "timestamp" : 1234567890123,
    "operation" : "CREATED|UPDATED|DELETED",
    "accountId" : "123456789012",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "thingName" : "MyThing",
    "versionNumber" : 1,
    "thingTypeName" : null,
    "attributes": {
        "attribute3": "value3",
        "attribute1": "value1",

```

```
        "attribute2": "value2"
    }
```

The payloads contain the following attributes:

**eventType**

Set to "THING\_EVENT".

**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.

**Thing Type Created/Deprecated/Undeleted**

The registry publishes the following event messages when thing types are created, deprecated, undeleted, or deleted:

- \$aws/events/thingType/<thingTypeName>/created
- \$aws/events/thingType/<thingTypeName>/updated
- \$aws/events/thingType/<thingTypeName>/deleted

The message contains the following example payload:

```
{
    "eventType" : "THING_TYPE_EVENT",
    "eventId" : "8827376c-4b05-49a3-9b3b-733729df7ed5",
    "timestamp" : 1234567890123,
    "operation" : "CREATED|UPDATED|DELETED",
    "accountId" : "123456789012",
    "thingTypeId" : "c530ae83-32aa-4592-94d3-da29879d1aac",
```

```
    "thingTypeName" : "MyThingType",
    "isDeprecated" : false|true,
    "deprecationDate" : null,
    "searchableAttributes" : [ "attribute1", "attribute2", "attribute3" ],
    "description" : "My thing type"
}
```

The payloads contain the following attributes:

**eventType**

Set to "THING\_TYPE\_EVENT".

**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.

**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.

**Thing Type Associated or Disassociated with a Thing**

The registry publishes the following event messages when a thing type is associated or disassociated with a thing.

- \$aws/events/thingTypeAssociation/thing/<thingName>/<typeName>

The messages contain the following example payload:

```
{
    "eventId" : "87f8e095-531c-47b3-aab5-5171364d138d",
    "eventType" : "THING_TYPE_ASSOCIATION_EVENT",
    "operation" : "CREATED|DELETED",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
```

```

    "thingName": "myThing",
    "thingTypeName" : "MyThingType",
    "timestamp" : 1234567890123,
}

```

The payloads contain the following attributes:  
eventId

A unique event ID (string).

eventType

Set to "THING\_TYPE\_ASSOCIATION\_EVENT".

operation

The operation that triggered the event. Valid values are:

- CREATED
- DELETED

thingId

The ID of the thing whose type association was changed.

thingName

The name of the thing whose type association was changed.

thingTypeName

The thing type associated with, or no longer associated with, the thing.

timestamp

The UNIX timestamp of when the event occurred.

#### Thing Group Created/Updated/Deleted

The registry publishes the following event messages when a thing group is created, updated, or deleted.

- \$aws/events/thingGroup/<groupName>/created
- \$aws/events/thingGroup/<groupName>/updated
- \$aws/events/thingGroup/<groupName>/deleted

The messages contain the following example payload:

```

{
    "eventType" : "THING_GROUP_EVENT",
    "eventId" : "87f8e095-531c-47b3-aab5-5171364d138d",
    "timestamp" : 1234567890123,
    "operation" : "CREATED|UPDATED|DELETED",
    "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 payloads contain the following attributes:

**eventType**

Set to "THING\_GROUP\_EVENT".

**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.

### Thing Added to or Removed from a Thing Group

The registry publishes the following event messages when a thing is added to or removed from a thing group.

- \$aws/events/thingGroupMembership/thingGroup/<thingGroupName>/thing/<thingName>/added
- \$aws/events/thingGroupMembership/thingGroup/<thingGroupName>/thing/<thingName>/removed

The messages contain the following example payload:

```
{
    "eventType" : "THING_GROUP_MEMBERSHIP_EVENT",
    "eventId" : "d684bd5f-6f6e-48e1-950c-766ac7f02fd1",
    "timestamp" : 1234567890123,
    "operation" : "ADDED|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"
}
```

The payloads contain the following attributes:

**eventType**

Set to "THING\_GROUP\_MEMBERSHIP\_EVENT".

**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.

### Thing Group Added to or Deleted from a Thing Group

The registry publishes the following event messages when a thing group is added to or removed from another thing group.

- \$aws/events/thingGroupHierarchy/thingGroup/<*parentThingGroupName*>/childThingGroup/<*childThingGroupName*>/added
- \$aws/events/thingGroupHierarchy/thingGroup/<*parentThingGroupName*>/childThingGroup/<*childThingGroupName*>/removed

The message contains the following example payload:

```
{  
    "eventType" : "THING_GROUP_HIERARCHY_EVENT",  
    "eventId" : "264192c7-b573-46ef-ab7b-489fc47da41",  
    "timestamp" : 1234567890123,  
    "operation" : "ADDED|REMOVED",  
    "accountId" : "123456789012",  
    "thingGroupId" : "8f82a106-6b1d-4331-8984-a84db5f6f8cb",  
    "thingGroupName" : "MyRootThingGroup",  
    "childGroupId" : "06838589-373f-4312-b1f2-53f2192291c4",  
    "childGroupName" : "MyChildThingGroup"  
}
```

The payloads contain the following attributes:

**eventType**

Set to "THING\_GROUP\_HIERARCHY\_EVENT".

**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.

## Jobs Events

The AWS IoT Jobs service 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. Use the [UpdateEventConfigurations](#) API to control the kinds of job events you receive.

Because it can take some time to cancel or delete a job, two messages are sent to indicate the start and end of a request. For example, when a cancellation request starts, a message is sent to the \$aws/events/job/jobID/cancellation\_in\_progress topic. When the cancellation request is complete,

a message is sent to the `$aws/events/job/jobID/canceled` topic. A similar process occurs for a job deletion request. Management and monitoring applications can subscribe to these topics to keep track of the status of jobs.

For more information about publishing and subscribing to MQTT topics, see [Message Broker for AWS IoT \(p. 244\)](#).

#### Job Completed/Canceled/Deleted

The AWS IoT Jobs service publishes a message on an MQTT topic when a job is completed, canceled, deleted, or when cancellation or deletion are in progress:

- `$aws/events/job/jobID/completed`
- `$aws/events/job/jobID/canceled`
- `$aws/events/job/jobID/deleted`
- `$aws/events/job/jobID/cancellation_in_progress`
- `$aws/events/job/jobID/deletion_in_progress`

The completed message contains the following example payload:

```
{
  "eventType": "JOB",
  "eventId": "7364ffd1-8b65-4824-85d5-6c14686c97c6",
  "timestamp": 1234567890,
  "operation": "completed",
  "jobId": "27450507-bf6f-4012-92af-bb8a1c8c4484",
  "status": "COMPLETED",
  "targetSelection": "SNAPSHOT|CONTINUOUS",
  "targets": [
    "arn:aws:iot:us-east-1:123456789012:thing/a39f6f91-70cf-4bd2-a381-9c66df1a80d0",
    "arn:aws:iot:us-east-1:123456789012:thinggroup/2fc4c0a4-6e45-4525-
a238-0fe8d3dd21bb"
  ],
  "description": "My Job Description",
  "completedAt": 1234567890123,
  "createdAt": 1234567890123,
  "lastUpdatedAt": 1234567890123,
  "jobProcessDetails": {
    "numberOfCanceledThings": 0,
    "numberOfRejectedThings": 0,
    "numberOfFailedThings": 0,
    "numberOfRemovedThings": 0,
    "numberOfSucceededThings": 3
  }
}
```

The canceled message contains the following example payload:

```
{
  "eventType": "JOB",
  "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
  "timestamp": 1234567890,
  "operation": "canceled",
  "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
  "status": "CANCELED",
  "targetSelection": "SNAPSHOT|CONTINUOUS",
  "targets": [
    "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
    "arn:aws:iot:us-east-1:123456789012:thinggroup/ThingGroup1-95c644d5-1621-41a6-9aa5-
ad2de581d18f"
  ],
}
```

```

    "description": "My job description",
    "createdAt": 1234567890123,
    "lastUpdatedAt": 1234567890123
}

```

The deleted message contains the following example payload:

```

{
    "eventType": "JOB",
    "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
    "timestamp": 1234567890,
    "operation": "deleted",
    "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
    "status": "DELETED",
    "targetSelection": "SNAPSHOT|CONTINUOUS",
    "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
    ],
    "description": "My job description",
    "createdAt": 1234567890123,
    "lastUpdatedAt": 1234567890123,
    "comment": "Comment for this operation"
}

```

The cancellation\_in\_progress message contains the following example payload:

```

{
    "eventType": "JOB",
    "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
    "timestamp": 1234567890,
    "operation": "cancellation_in_progress",
    "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
    "status": "CANCELLATION_IN_PROGRESS",
    "targetSelection": "SNAPSHOT|CONTINUOUS",
    "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
    ],
    "description": "My job description",
    "createdAt": 1234567890123,
    "lastUpdatedAt": 1234567890123,
    "comment": "Comment for this operation"
}

```

The deletion\_in\_progress message contains the following example payload:

```

{
    "eventType": "JOB",
    "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
    "timestamp": 1234567890,
    "operation": "deletion_in_progress",
    "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
    "status": "DELETION_IN_PROGRESS",
    "targetSelection": "SNAPSHOT|CONTINUOUS",
    "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
    ]
}

```

```
    "arn:aws:iot:us-east-1:123456789012:thinggroup/  
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"  
],  
"description": "My job description",  
"createdAt": 1234567890123,  
"lastUpdatedAt": 1234567890123,  
"comment": "Comment for this operation"  
}
```

### Job Execution Terminal Status

The AWS IoT Jobs service 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
- \$aws/events/jobExecution/*jobID*/timed\_out
- \$aws/events/jobExecution/*jobID*/removed
- \$aws/events/jobExecution/*jobID*/deleted

The message contains the following example payload:

```
{  
  "eventType": "JOB_EXECUTION",  
  "eventId": "cca89fa5-8a7f-4ced-8c20-5e653afb3572",  
  "timestamp": 1234567890,  
  "operation": "succeeded|failed|rejected|canceled|removed|timed_out",  
  "jobId": "154b39e5-60b0-48a4-9b73-f6f8dd032d27",  
  "thingArn": "arn:aws:iot:us-east-1:123456789012:myThing/6d639fbc-8f85-4a90-924d-a2867f8366a7",  
  "status": "SUCCEEDED|FAILED|REJECTED|CANCELED|REMOVED|TIMED_OUT",  
  "statusDetails": {  
    "key": "value"  
  }  
}
```

## 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* – A client connected to the message broker.
- \$aws/events/presence/disconnected/*clientId* – A client disconnected from the message broker.

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.

### clientInitiatedDisconnect

True if the client initiated the disconnect. Otherwise, false. Found in disconnection messages only.

### disconnectReason

The reason why the client is disconnecting. Found in disconnect messages only. The following table contains valid values.

| Disconnect Reason           | Description                                                                                                                                                                                                                                                    |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AUTH_ERROR                  | The client failed to authenticate or authorization failed.                                                                                                                                                                                                     |
| CLIENT_INITIATED_DISCONNECT | The client indicates that it will disconnect. The client can do this by sending either a MQTT DISCONNECT control packet or a Close frame if the client is using a WebSocket connection.                                                                        |
| CLIENT_ERROR                | The client did something wrong that causes it to disconnect. For example, a client will be disconnected for sending more than 1 MQTT CONNECT packet on the same connection or if the client attempts to publish with a payload that exceeds the payload limit. |
| CONNECTION_LOST             | The client-server connection is cut off. This can happen during a period of high network latency or when the internet connection is lost.                                                                                                                      |
| DUPLICATE_CLIENTID          | The client is using a client ID that is already in use. In this case, the client that is already connected will be disconnected with this disconnect reason.                                                                                                   |
| FORBIDDEN_ACCESS            | The client is not allowed to be connected. For example, a client with a denied IP address will fail to connect.                                                                                                                                                |
| MQTT_KEEP_ALIVE_TIMEOUT     | If there is no client-server communication for 1.5x of the client's keep-alive time, the client is disconnected.                                                                                                                                               |
| SERVER_ERROR                | Disconnected due to unexpected server issues.                                                                                                                                                                                                                  |
| SERVER_INITIATED_DISCONNECT | Server intentionally disconnects a client for operational reasons.                                                                                                                                                                                             |
| THROTTLED                   | The client is disconnected for exceeding a throttling limit.                                                                                                                                                                                                   |
| WEBSOCKET_TTL_EXPIRATION    | The client is disconnected because a WebSocket has been connected longer than its time-to-live value.                                                                                                                                                          |

### **eventType**

The type of event. Valid values are `connected` or `disconnected`.

### **ipAddress**

The IP address of the connecting client. This can be in IPv4 or IPv6 format. Found in connection messages only.

### **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.

### **versionNumber**

The version number for the lifecycle event. This is a monotonically increasing long integer value for each client ID connection. The version number can be used by a subscriber to infer the order of lifecycle events.

#### **Note**

The connect and disconnect messages for a client connection have the same version number.

The version number might skip values and is not guaranteed to be consistently increasing by 1 for each event.

If a client is not connected for approximately one hour, the version number is reset to 0. For persistent sessions, the version number is reset to 0 after a client has been disconnected longer than the configured time-to-live (TTL) for the persistent session.

A connect message has the following structure.

```
{
  "clientId": "186b5",
  "timestamp": 1573002230757,
  "eventType": "connected",
  "sessionIdentifier": "a4666d2a7d844ae4ac5d7b38c9cb7967",
  "principalIdentifier": "12345678901234567890123456789012",
  "ipAddress": "192.0.2.0",
  "versionNumber": 0
}
```

A disconnect message has the following structure.

```
{
  "clientId": "186b5",
  "timestamp": 1573002340451,
  "eventType": "disconnected",
  "sessionIdentifier": "a4666d2a7d844ae4ac5d7b38c9cb7967",
  "principalIdentifier": "12345678901234567890123456789012",
  "clientInitiatedDisconnect": true,
  "disconnectReason": "CLIENT_INITIATED_DISCONNECT",
  "versionNumber": 0
}
```

}

## Handling Client Disconnections

The best practice is to always have a wait state implemented for lifecycle events, including Last Will and Testament (LWT) messages. When a disconnect message is received, your code should wait a period of time and verify a device is still offline before taking action. One way to do this is by using [SQS Delay Queues](#). When a client receives a LWT or a lifecycle event, you can enqueue a message (for example, for 5 seconds). When that message becomes available and is processed (by Lambda or another service), you can first check if the device is still offline before taking further action.

## 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/ABCDEFGHIJKLMNPQRSTUVWXYZ:some-user/  
ABCDEFGHIJKLMNPQRSTUVWXYZ: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.

# Alexa Voice Service Integration for AWS IoT

Alexa Voice Service (AVS) Integration for AWS IoT is a new feature that cost-effectively brings Alexa Voice to any connected device without incurring [messing costs](#). AVS for AWS IoT reduces the cost and complexity of integrating Alexa. This feature leverages AWS IoT to offload intensive computational and memory audio tasks from the device to the cloud. Because of the resulting reduction in the engineering bill of materials (eBoM) cost, device makers can now cost-effectively bring Alexa to resource-constrained IoT devices and make it possible for consumers to talk directly to Alexa in parts of their home, office, or hotel rooms for an ambient experience.

Currently, smart home IoT devices are built with low-cost microcontrollers (MCU) that have limited memory to run real-time operating systems. Previously, AVS solutions for Alexa built-in products required expensive application processor-based devices with more than 50 MB memory running on Linux or Android. These expensive hardware requirements made it cost-prohibitive to integrate Alexa Voice on resource-constrained IoT devices. AVS for AWS IoT enables Alexa built-in functionality on MCUs, such as the ARM Cortex M class with less than 1 MB embedded RAM. To do so, AVS offloads memory and compute tasks to a virtual Alexa built-in device in the cloud. This reduces eBoM cost by up to 50 percent.

For more information about the ARM Cortex M class, see [ARM Cortex-M](#) on Wikipedia. For more information about hardware requirements for Alexa built-in products, see [Sizing Up CPU, Memory, and Storage for Your Alexa Built-in Device](#) on the Amazon Alexa developer portal.

## Note

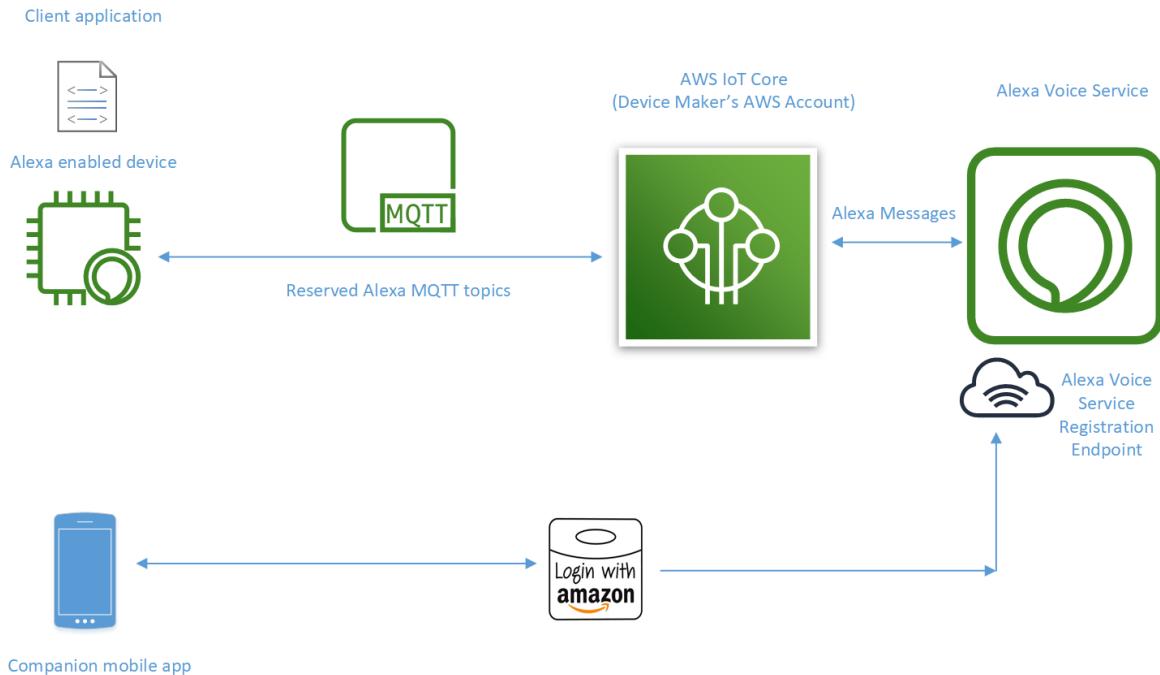
AVS for AWS IoT is available in all AWS Regions where AWS IoT is available except in the China (Beijing and Ningxia) Regions. For the current list of AWS Regions, see the [AWS Region Table](#).

AVS for AWS IoT has three components:

- A set of reserved MQTT topics to transfer audio messages between Alexa enabled devices and AVS.
- A virtual Alexa enabled device in the cloud that shifts tasks related to media retrieval, audio decoding, audio mixing, and state management from the physical device to the virtual device.
- A set of APIs that support receiving and sending messages over the reserved topics, interfacing with the device microphone and speaker, and managing device state.

The following diagram illustrates how these components work together. It also demonstrates how device makers use the Login with Amazon service to authenticate AVS.

## Alexa Voice Service (AVS) Integration for AWS IoT Core



Device manufacturers have two options to get started with AVS Integration for AWS IoT .

- **Development kits** – Development kits launched by our partners make it easy to get started. The [NXP i.MX RT 106 A](#) and [Qualcomm Home Hub 100 Development Kit for Amazon AVS](#) are the first two kits available on the market. You can find them on [Development Kits for AVS](#). The kits include out-of-the box connectivity to AWS IoT, AVS qualified Audio Algorithms for Far-Field voice pickup, Echo Cancellation, Alexa Wake Word, and AVS for AWS IoT application code. You can use the feature application code to quickly prototype a device and port the implementation to your chosen MCU design for testing and device production when you're ready.
- **Custom device-side application code** – Developers can also write a custom AVS for AWS IoT application by using the publicly available API. Documentation for this API is available on the [AVS developer page](#). You can download the FreeRTOS and AWS IoT Device SDK from the FreeRTOS console (<https://console.aws.amazon.com/freertos/>) or [GitHub](#).

To get started with an NXP i.MX 106A development kit, see [Getting Started with Alexa Voice Service \(AVS\) Integration for AWS IoT on an NXP Device](#).

## Getting Started with Alexa Voice Service Integration for AWS IoT on an NXP Device

The NXP i.MX 106A development kit enables you to preview Alexa Voice Service (AVS) Integration for AWS IoT using a preconfigured NXP account. After you preview the functionality with the NXP account, you need to customize the firmware, application source code, and the NXP mobile application provided with the kit to use your own account. This topic walks you through the steps to preview with the preconfigured account and to customize your device with your own account.

## Topics

- [Preview AVS Integration for AWS IoT with a Preconfigured NXP Account \(p. 700\)](#)
- [Use Your AWS and Alexa Voice Service Developer Accounts to Set Up AVS for AWS IoT \(p. 703\)](#)

# Preview AVS Integration for AWS IoT with a Preconfigured NXP Account

## Prerequisites

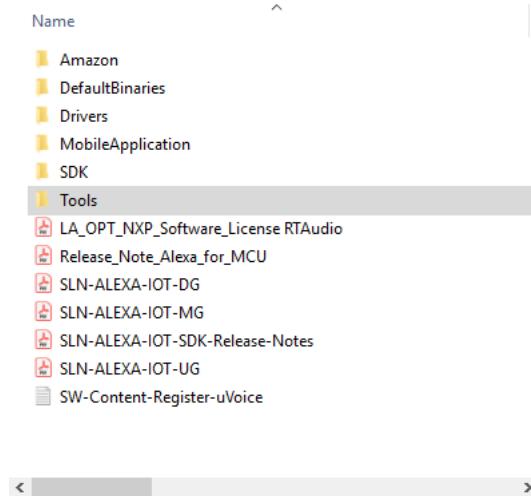
To follow these steps, you need the following resources.

- [NXP i.MX 106A development kit](#)
- A Mac, Windows 7/10, or Linux computer
- A serial driver that works with your computer
- An Android or iOS mobile device
- Android Debug Bridge (ADB)
- An [Amazon Alexa account](#)

## Install and Boot the Development Kit

1. Activate the device kit. The kit comes with a Get Started Onboarding card. This card contains instructions to activate the kit and acquire the necessary software package and the reference design files. The software package contains the SDK source code and the companion Android application (`VoiceCompanionApp.apk`). If you're using an iOS device, you must request access to the TestFlight iOS application from your local NXP representative.

After you download and extract the software package, you see the following file structure.



The `MobileApplication` folder contains the Android mobile application.

The `Tools` folder contains scripts that you use to configure your device.

You can also find the following documentation:

- `SLN-ALEXA-IOT-DG.pdf` in the [NXP Developer Guide](#)

- [SLN-ALEXA-IOT-MG.pdf](#) in the *NXP Migration Guide*
  - [SLN-ALEXA-IOT-UG.pdf](#) in the *NXP User Guide*
2. Boot the device kit. The USB splitter cable that comes with the kit has power and data connections.
- a. Connect both of the USB-A connections to your computer.
  - b. Connect the USB-C connector to your kit. Your configuration might look the following image.



When the board has power, the D1 LED lights up and displays green. The D2 LED (closest to the speaker) indicates the state of the device. When the device turns on, this LED flashes multiple colors. When the device runs the application code, the D2 LED blinks yellow. When initialization of the device is complete, the D2 LED displays orange to indicate that it's waiting for Wi-Fi credentials to be added to the device.

The NXP User Guide contains a description of the device's physical controls.



3. Use the terminal application to understand how the client application on the device works. You can also use it to debug and make sure that the device is in the correct state.

Enter the following commands to get started with the application:

- `help` – Displays a list of the available commands.
- `enable_usb_log` – Enables logging on the device. This helps you understand what the application is doing.
- `logs` – Displays the last few lines of the logs. At this point, the logs should indicate that the device is waiting for Wi-Fi credentials.

## Connect the Device to AWS IoT and Amazon Alexa

1. Install the appropriate mobile application. If you're using an Android device, you can use ADB in the terminal to install the `voiceCompanionApp.apk` file. If you're using an iOS device, use the TestFlight application.
2. Provision the Wi-Fi credentials. You can provision the Wi-Fi credentials by using either the companion mobile application or the terminal.
  - a. Follow these steps to provision the Wi-Fi credentials by using the mobile application.
    - i. When the device boots up from its factory new state, it creates a Wi-Fi access point. Open the companion mobile application, and choose **WIFI PROVISION** to connect to this access point.
    - ii. Choose a network from the list.
    - iii. Enter your password, and choose **Send** to transmit the credentials to the device kit.
  - b. In the terminal application, enter the following command.

```
setup YourWiFiNetworkName YourWiFiNetworkPassword
```

- c. Choose **Enter**.
3. Authenticate the device kit and connect to AWS IoT and Amazon Alexa. Make sure that your mobile device and the device kit are using the same Wi-Fi network so that the two devices can communicate.

To do so, press the **Discover** button in the mobile application. When discovery is complete, you see a list of serial numbers for the available device kits near you. If more than one is available, you can confirm your kit's unique serial number by entering **serial\_number** in the terminal application.
4. Choose the device kit that to use. The mobile application uses the Login with Amazon service to prompt you for your Amazon user credentials. The device kit begins to register the device with AVS. The D2 LED displays purple to indicate that device registration has started.

**Note**

If your mobile device already has the Amazon shopping application installed, the mobile companion application automatically uses the Amazon account that is logged into the shopping application.

After you sign in to your Amazon account, the companion application performs the following steps:

1. The device kit receives the access token from the Login with Amazon service and begins to connect the device to AWS IoT and AVS. The mobile application displays a completion percentage. The D2 LED on the device displays orange.
2. The D1 LED blinks green every 500 milliseconds until the device connects to AWS IoT.
3. When the device connects to AWS IoT, the device kit begins to connect the device to AVS. The D1 LED blinks green every 250 milliseconds.
4. When the device connects to AVS, the color of the device kit's serial number changes to yellow in the mobile application. The mobile application displays a **Complete** message. The D1 LED turns off, and the device plays a chime. The device is now connected to NXP's AWS IoT account.

You can try a few Alexa Voice commands, such as the following examples:

- "Alexa, what's the weather like?"
- "Alexa, play a news briefing."
- "Alexa, play music."

## Use Your AWS and Alexa Voice Service Developer Accounts to Set Up AVS for AWS IoT

### Prerequisites

For this procedure, you need the following resources.

- [NXP i.MX 106A development kit](#)
- [MCUXPresso v10.3.1](#)
- [Segger J-Link debug probe](#)
- A Mac computer
- An Android mobile device
- [Android Studio](#)

- Android Debug Bridge
- An [Amazon Alexa account](#)
- An [AWS account](#)

## Configure Your AWS IoT Account and Provision Your Device

1. Generate credentials to authenticate with AWS IoT. All devices must have a device certificate, private key, and root CA certificate installed in their firmware to communicate with AWS IoT. Follow the instructions in [Register a Device in the Registry](#) to register your device with AWS IoT. For more information about X.509 certificates, see [X.509 Client Certificates](#).
2. Navigate to the root folder and open the NXP Developer Guide. Follow the instructions in "Section 9 (Filesystem)" to convert your client certificates and keys to a binary format that the NXP filesystem can read.

Use the following script and commands to convert the certificates and keys. The `bin_dump.py` script is in the `Tools\Ivaldi.zip\Scripts\sln_alexa_iot_utils` folder.

```
python3 bin_dump.py CertificateName-certificate.pem.crt CertificateName-
certificate.pem.crt.bin
```

```
python3 bin_dump.py CertificateName-private.pem.key CertificateName-private.pem.key.bin
```

For information about options for generating credentials for production devices at scale, see [Device Provisioning](#).

### To set up your device in the Alexa Voice Service Developer console

1. Navigate to the root folder and open the NXP Migration Guide.
2. To get the MD5 and SHA256 signatures that you need to create an AVS API key, follow the instructions in "Section 2. Creating a Keystore for Android AVS Companion Application" in the *NXP Migration Guide*. (This includes navigating to the [Alexa Voice Service Developer Console](#).)
3. To create a [Login with Amazon](#) security profile and create an AVS product, follow the instructions in "Section 3" in the *NXP Migration Guide*.

### To rebuild the Android mobile app

1. Open the `VoiceCompanionApp` project in Android Studio. The `VoiceCompanionApp.apk` file is included in the NXP software package.
2. In Android Studio, add the API key that you generated in the previous procedure to the `api_key.txt` file in the `assets` folder. This synchronizes the companion mobile application with your AVS security profile. The application uses the API key when you log in by using the Login with Amazon service. When the application gets an authorization code from Amazon, it transfers the code to the device firmware. The device then obtains access and refresh tokens from the Login with Amazon service to make calls to AVS.
3. In Android Studio, choose **Build Project** and **Generate a Signed Bundle/APK**. For more information, see "Section 4. Building and Generating the Mobile Application" in the *NXP Migration Guide*. For more information about authorizing with Alexa in this way, see [Authorize from a Companion App \(Android/iOS\)](#).
4. Install the updated mobile application to your mobile device. Set up ADB on your Mac computer and install the updated Android APK file to your mobile phone by using the terminal console. If you installed the mobile application to preview the AVS for AWS IoT service with the preconfigured NXP account, you must uninstall and then reinstall the application.

### To update the client application with your AWS credentials

1. Follow the instructions in Section 3 and Section 4 in the *NXP Developer Guide* to make the required Segger J-Link driver modifications. These instructions also explain how to import the `Bootloader`, `ais_demo`, and `bootstrap` projects by using the MCUXPresso IDE.
2. Follow the instructions for updating the source code and rebuilding the application in "Section 7. RT106A Firmware Changes" in the *NXP Migration Guide*. We recommend that you use MCUXPresso v10.3.1.

These source code changes enable the device firmware to communicate with your AWS account and your AVS product instead of the default NXP account.

### To set up and connect your device to AWS IoT and Amazon Alexa

1. Reset the device to factory defaults. If you previewed the AVS for AWS IoT service with the preconfigured NXP account, you must reset the device firmware before you log in again with the companion mobile application. This ensures that the mobile application and the device use your security profile. Make sure that the device is connected to your computer. Push **SW1** for 10 seconds to initiate the factory reset.

The *NXP User Guide* that is included in the NXP software package contains a description of the device's physical controls.

2. Reprogram the firmware to use the certificate and key that you generated for your device. Add the key and certificate to the updated `Bootstrap` and `Ais_demo` binaries that you created in the previous procedure. We also recommend reprogramming the firmware with the default `Intelligent_toolbox`, `app_crt` and `CA_crt` binaries.

For instructions, see "Section 5. Building and Programming" in the *NXP Developer Guide*.

3. Follow the instructions in the previous [Connect the Device to AWS and Amazon Alexa \(p. 702\)](#) procedure. This connects you to AVS for AWS IoT with your own security profile and credentials.

# AWS IoT Device and Mobile SDKs

## Contents

- [AWS Mobile SDK for Android \(p. 706\)](#)
- [Arduino Yún SDK \(p. 706\)](#)
- [AWS IoT Device SDK for Embedded C \(p. 706\)](#)
- [AWS IoT C++ Device SDK \(p. 707\)](#)
- [AWS Mobile SDK for iOS \(p. 707\)](#)
- [AWS IoT Device SDK for Java \(p. 707\)](#)
- [AWS IoT Device SDK for JavaScript \(p. 707\)](#)
- [AWS IoT Device SDK for Python \(p. 708\)](#)

The AWS IoT Device and Mobile SDKs help you to easily and quickly connect your devices to AWS IoT. The AWS IoT Device and Mobile 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. For the AWS SDKs used to access AWS APIs, see [AWS SDKs and Tools](#).

### Note

Fleet provisioning (beta) is currently supported only in the AWS IoT Device SDK for Embedded C.

## 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 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 Device Shadow 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 require 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 shadow support. You can access shadows by using HTTP methods, including GET, UPDATE, and DELETE. The SDK also supports a simplified shadow access model, which allows developers to exchange data with 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 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](#)

# Troubleshooting AWS IoT

The following information might help you troubleshoot common issues in AWS IoT.

## Tasks

- [Diagnosing Connectivity Issues \(p. 709\)](#)
- [Diagnosing Rules Issues \(p. 709\)](#)
- [Diagnosing Problems with Shadows \(p. 710\)](#)
- [Diagnosing Salesforce IoT Input Stream Action Issues \(p. 711\)](#)
- [Troubleshooting Aggregation Queries for the Fleet Indexing Service \(p. 712\)](#)
- [AWS IoT Device Defender Troubleshooting Guide \(p. 713\)](#)
- [AWS IoT Errors \(p. 716\)](#)

## Diagnosing Connectivity Issues

### Authentication

How do my devices authenticate AWS IoT endpoints?

Add the AWS IoT CA certificate to your client's trust store. Refer to the documentation on [Server Authentication in AWS IoT Core](#) and then follow the links to download the appropriate CA certificate.

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
```

For more information about using `openssl s_client`, see [OpenSSL s\\_client documentation](#).

### 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.

## Diagnosing Rules Issues

The best way to debug issues you are having with rules is to use CloudWatch Logs. 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. For more information, see [Monitoring with CloudWatch Logs \(p. 209\)](#).

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 \(p. 213\)](#):

```
{
    "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 \(p. 211\)](#):

```
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” \(p. 231\)](#).

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 Shadows

### Diagnosing Shadows

| Issue                                                                                              | Troubleshooting Guidelines                                                                                                                                              |
|----------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A device's 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 <a href="#">Device Shadow Document Syntax</a> . |
| I submitted correct JSON, but none or only parts of it are stored in the device's shadow document. | Be sure you are following the JSON formatting guidelines. Only JSON fields in the desired and                                                                           |

| Issue                                                                                                                                                            | Troubleshooting Guidelines                                                                                                                                                                                                                                                                                                         |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                                                  | reported sections are stored. JSON content (even if formally correct) outside of those sections is ignored.                                                                                                                                                                                                                        |
| I received an error that the device's shadow exceeds the allowed size.                                                                                           | The device's shadow supports 8 KB of data only. Try shortening field names inside of your JSON document or simply create more shadows by creating more things. A device can have an unlimited number of things/shadows associated with it. The only requirement is that each thing name must be unique in your account.            |
| When I receive a device's shadow, it is larger than 8 KB. How can this happen?                                                                                   | Upon receipt, the AWS IoT service adds metadata to the device's 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 device's 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. |
| My device can publish and subscribe on the corresponding shadow topics, but when I attempt to update the 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 Device Shadow 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?

See the [Monitoring with CloudWatch Logs \(p. 209\)](#) section. After you have activated the logs, you can 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.

## Troubleshooting Aggregation Queries for the Fleet Indexing Service

If you are having type mismatch errors, you can use CloudWatch Logs to troubleshoot the problem. CloudWatch Logs must be enabled before logs are written by the Fleet Indexing service. For more information, see [CloudWatch Logs](#).

When you make aggregation queries on non-managed fields, you can only specify a field you defined in the `customFields` argument passed to `UpdateIndexingConfiguration` or `update-indexing-configuration`. If the field value is inconsistent with the configured field data type, this value is ignored when you perform an aggregation query.

The Fleet Indexing service emits an error log to CloudWatch Logs when a field cannot be indexed because of a mismatched type. The error log contains the field name, the value that could not be converted, and the thing name for the device. The following is an example error log:

```
{  
  "timestamp": "2017-02-20 20:31:22.932",
```

```
"logLevel": "ERROR",
"traceId": "79738924-1025-3a00-a669-7bec69f7f07a",
"accountId": "000000000000",
"status": "SucceededWithIssues",
"eventType": "IndexingCustomFieldFailed",
"thingName": "thing0",
"failedCustomFields": [
  {
    "Name": "attributeName1",
    "Value": "apple",
    "ExpectedType": "String"
  },
  {
    "Name": "attributeName2",
    "Value": "2",
    "ExpectedType": "Boolean"
  }
]
```

If a device has been disconnected for approximately an hour, the connectivity status timestamp value might be missing. For persistent sessions, the value might be missing after a client has been disconnected longer than the configured time-to-live (TTL) for the persistent session. The connectivity status data is indexed only for connections where the client ID has a matching thing name. (The client ID is the value used to connect a device to AWS IoT Core.)

## AWS IoT Device Defender Troubleshooting Guide

### General

Q: Are there any prerequisites for using AWS IoT Device Defender?

A: If you want to use device-reported metrics, you must first deploy an agent on your AWS IoT connected devices or device gateways. Devices must provide a consistent client identifier or thing name.

### Audit

Q: I enabled a check and my audit has been showing "In-Progress" for a long time. Is something wrong? When can I expect results?

A: When a check is enabled, data collection starts immediately. However, if your account has a large amount of data to collect (certificates, things, policies, and so on), the results of the check might not be available for some time after you have enabled it.

### Detect

Q: How do I know the thresholds to set in an AWS IoT Device Defender security profile behavior?

A: Start by creating a security profile behavior with low thresholds and attach it to a thing group that contains a representative set of devices. You can use AWS IoT Device Defender to view the current metrics, and then fine-tune the device behavior thresholds to match your use case.

Q: I created a behavior, but it is not triggering a violation when I expect it to. How should I fix it?

A: When you define a behavior, you are specifying how you expect your device to behave normally. For example, if you have a security camera that only connects to one central server on TCP port 8888, you don't expect it to make any other connections. To be alerted if the camera makes a connection on another port, you define a behavior like this:

```
{  
  "name": "Listening TCP Ports",  
  "metric": "aws:listening-tcp-ports",  
  "criteria": {  
    "comparisonOperator": "in-port-set",  
    "value": {  
      "ports": [ 8888 ]  
    }  
  }  
}
```

If the camera makes a TCP connection on TCP port 443, the device behavior would be violated and an alert would be triggered.

Q: One or more of my behaviors are in violation. How do I clear the violation?

A: Alarms clear after the device returns to expected behavior, as defined in the behavior profiles. Behavior profiles are evaluated upon receipt of metrics data for your device. If the device doesn't publish any metrics for more than two days, the violation event is set to `alarm-invalidated` automatically.

Q: I deleted a behavior that was in violation, but how do I stop the alerts?

A: Deleting a behavior stops all future violations and alerts for that behavior. Earlier alerts must be drained from your notification mechanism. When you delete a behavior, the record of violations of that behavior is retained for the same time period as all other violations in your account.

## Device Metrics

Q: I'm submitting metrics reports that I know violate my behaviors, but no violations are being triggered. What's wrong?

A: Check that your metrics reports are being accepted by subscribing to the following MQTT topics:

```
$aws/things/THING_NAME/defender/metrics/FORMAT/rejected  
$aws/things/THING_NAME/defender/metrics/FORMAT/accepted
```

where `THING_NAME` is the name of the thing reporting the metric and `FORMAT` is either "json" or "cbor", depending on the format of the metrics report submitted by the thing.

After you have subscribed, you should receive messages on these topics for each metric report submitted. A `rejected` message indicates that there was a problem parsing the metric report. An error message is included in the message payload to help you correct any errors in your metric report. An `accepted` message indicates the metric report was parsed properly.

Q: What happens if I send an empty metric in my metric report?

A: An empty list of ports or IP addresses is always considered in conformity with the corresponding behavior. If the corresponding behavior was in violation, the violation is cleared.

Q: Why do my device metric reports contain messages for devices that aren't in the AWS IoT registry?

If you have one or more security profiles attached to all things or to all unregistered things, AWS IoT Device Defender includes metrics from unregistered things. If you want to exclude metrics from unregistered things, you can attach the profiles to all registered devices instead of all devices.

Q: I'm not seeing messages from one or more unregistered devices even though I applied a security profile to all unregistered devices or all devices. How can I fix it?

Verify that you are sending a well-formed metrics report using one of the supported formats. For information, see [Device Metrics Document Specification \(p. 657\)](#). Verify that the unregistered devices

are using a consistent client identifier or thing name. Messages reported by devices are rejected if the thing name contains control characters or if the thing name is longer than 128 bytes of UTF-8 encoded characters.

Q: What happens if an unregistered device is added to the registry or a registered device becomes unregistered?

A: If a device is added to or removed from the registry:

- You see two separate violations for the device (one under its registered thing name, one under its unregistered identity) if it continues to publish metrics for violations. Active violations for the old identity stop appearing after two days, but are available in violations history for up to 14 days.

Q: Which value should I supply in the report ID field of my device metrics report?

A: Use a unique value for each metric report, expressed as a positive integer. A common practice is to use a [Unix epoch timestamp](#).

Q: Should I create a dedicated MQTT connection for AWS IoT Device Defender metrics?

A: A separate MQTT connection is not required.

Q: Which client ID should I use when connecting to publish device metrics?

For devices (things) that are in the AWS IoT registry, use the registered thing name. For devices that are not in the AWS IoT registry, use a consistent identifier when you connect to AWS IoT. This practice helps match the violations to the thing name.

Q: Can I publish metrics for a device with a different client ID?

It is possible to publish metrics on behalf of another thing. You can do this by publishing the metrics to the AWS IoT Device Defender reserved topic for that device. For example, Thing-1 would like to publish metrics for itself and also on behalf of Thing-2. Thing-1 collects its own metrics and publishes them on the MQTT topic:

```
$aws/things/Thing-1/defender/metrics/json
```

Thing-1 then obtains metrics from Thing-2 and publishes those metrics on the MQTT topic:

```
$aws/things/Thing-2/defender/metrics/json
```

Q: How many security profiles and behaviors can I have in my account?

A: See [AWS IoT Device Defender Endpoints and Quotas](#).

Q: What does a prototypical target role for an alert target look like?

A: A role that allows AWS IoT Device Defender to publish alerts on an alert target (SNS topic) requires two things:

- A trust relationship that specifies `iot.amazonaws.com` as the trusted entity.
- An attached policy that grants AWS IoT permission to publish on a specified SNS topic. For example:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": "<sns-topic-arn>"
        }
    ]
}
```

# 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.     |

## Device 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.       |

# AWS IoT Quotas

For AWS IoT Core quotas information, see [AWS IoT Core Endpoints and Quotas](#) in the *AWS General Reference*.