AWS IoT 1-Click

Developer Guide

AWS IoT 1-Click: Developer Guide

Table of Contents

What Is AWS IoT 1-Click?

AWS IoT 1-Click makes it easy for enterprise customers to incorporate simple IoT devices into their workflows without having to manufacture devices, write firmware, or configure them for secure connectivity. Our manufacturing partners create devices that can securely connect to AWS IoT right out of the box. These devices can trigger AWS Lambda functions that are written in languages like Java, Python, and C#. The Lambda functions can implement business logic on their own or trigger actions in the AWS Cloud or on-premises.

AWS IoT 1-Click aims to simplify the Internet of Things for customers by abstracting as much detail related to the device hardware and firmware as possible. This makes it possible to view AWS IoT 1-Click devices as software components hosted in the AWS Cloud. As with any other software component, these devices conform to well-defined interfaces. AWS IoT 1-Click has interfaces defined per device type. You can use these interfaces to build and base your applications on.

With AWS IoT 1-Click, you can group devices by function, location, or other criteria. This logical grouping of devices is called a "*project*" in AWS IoT 1-Click. You can use projects to associate groups of devices with Lambda functions for desired actions.

Projects contain templates that specify what type of devices are used, what Lambda functions they invoke, and what optional attributes, such as contextual data for location or function, are defined for these devices.

Once the project is created and templates defined, you can add placements within the project - each of which follows the template and specifies actual devices by their serial numbers and attribute values (key/value pairs) that make sense to the specific location or function for that particular placement.

AWS IoT 1-Click Components

Claim

Refers to the process of associating an AWS IoT 1-Click device with an AWS account using the AWS IoT 1-Click console ,AWS IoT 1-Click mobile app , or the AWS IoT 1-Click API.

Claim code

A value used to claim a number of AT&T LTE-M buttons at once (that is, in bulk). You can also use device IDs to claim devices. See the **Device ID** entry.

Device

A physical device, such as the AWS IoT Enterprise Button or the AT&T LTE-M Button.

Device attributes

Either default or custom data associated with a particular device in the form of key-value pairs. Default attributes are derived from the placement. See the **placement** entry.

Device ID

All devices have a device ID, such as a device serial number (DSN). A device ID can be used to to register an AWS IoT 1-Click device with AWS IoT 1-Click. A claim code is not the same as a device ID. See the **Claim code** entry.

Placement

A group of one or more templates representing devices (for example, a room containing two templatized buttons). To populate a placement, you use the AWS IoT 1-Click console or the AWS IoT 1-Click mobile app to choose templatized devices.

Placement name

The name of the placement, which often includes a geographic location or object ID (for example, `Room 217`, `North Dumpster`, or `Container 314`).

Project

A named group of zero or more placements (containing templatized devices).

Project name

A descriptive name for a group of placements (for example, "Meeting Room Satisfaction" or "Charter Container Pickup").

Template

Used to provide default behavior and default attributes for a group of devices. A physical device uses a particular template to inherit the properties of that template: its Lambda function and default device attributes. A template defines the behavior and default attributes for a class of device(s) in a placement. A project can have more than one template.

Unclaim

The process of disassociating an AWS IoT 1-Click device from an AWS account. For example, a person who wants to lend an AWS IoT 1-Click device should first disassociate the device from the AWS account so that the new user can associate the device with their own AWS account.

How AWS IoT 1-Click Works

Here is the AWS IoT 1-Click workflow:

1. Choose from a set of supported devices.
2. Associate AWS Lambda functions with devices to trigger actions. You can use one of your own Lambda functions or one of the predefined functions provided by the service.
3. Physically deploy your devices and use the AWS IoT 1-Click console, AWS IoT 1-Click mobile app, or AWS IoT 1-Click API to enable them.
4. Get information about device status and usage by using ready-made AWS IoT 1-Click reports or building your own.

AWS IoT 1-Click Devices

AWS IoT 1-Click-supported devices:

- Are ready-made. Customers do not need to design or manufacture them.
- Can be added to AWS IoT 1-Click accounts by using the claim feature (p. 6).
- Are pre-provisioned with certificates at the point of manufacture and configured to connect securely to AWS IoT. You don't need to spend time installing certificates for AWS IoT 1-Click devices.
- Each AWS IoT 1-Click device *type* emits events in a standard format defined by AWS IoT 1-Click. For example, all AWS IoT 1-Click devices of the `button` type have the same event format, regardless of the manufacturer.
- Have a device type and a product type. The device type indicates the format of events emitted by the device and the device methods it supports. For more information, see AWS IoT 1-Click Programming Model (p. 10). The product type provides manufacturer and the branding details. For example, if the device type is `button`, the product type might be AT&T LTE-M Button.

Important
AWS IoT 1-Click-supported devices are configured in the factory to connect to a particular AWS Region. These are called *device regions*. This association of a device to a *device region* is required to ensure that devices connect securely to AWS IoT without additional user input. For this reason, the device region cannot be changed.
Events emitted by AWS IoT 1-Click devices are always routed through the pre-configured device region, so you can access device-related Amazon CloudWatch Logs and AWS CloudTrail metrics in this same AWS Region. The device region is also where enabled devices are billed.
Placement, template, and project data is stored in the AWS Region associated with your account.This region can be different from the device region.

For information about AWS IoT 1-Click supported devices, including how to purchase and claim (p. 3) them, see AWS IoT 1-Click Appendix (p. 39).

Claiming Devices

When AWS IoT 1-Click devices leave the factory, they are not associated with an AWS customer account. Customers must go through a claim process to use the devices in their accounts. There are two ways to claim devices:

- **Using a claim code**: If you receive a claim code (in the format C-*xxxxx*) from your point of purchase, you can enter it into the AWS IoT 1-Click console or the AWS IoT 1-Click mobile app to claim devices pertaining to a single order. Not all devices, including the AWS IoT Enterprise Button, can be claimed using a claim code.
- **Using a device ID**: You can use the device ID (the device serial number, also known as the DSN) to claim devices through the AWS IoT 1-Click console or the AWS IoT 1-Click mobile app. All AWS IoT 1-Click devices can be claimed using a device ID.

For more information about how to claim devices, see AWS IoT 1-Click Appendix (p. 39) and Claiming Devices (p. 6).

Projects, Templates, and Placements

Devices can be organized by function, location, or by any other criteria. This logical grouping of devices is called a project. You can use projects to associate groups of devices with Lambda functions.

Projects contain templates that specify which type of devices are used, which Lambda functions they invoke, and attribute names to hold contextual data like location or function.

Once the project is created and templates defined, you can add placements in the project. Placements follow the template and specify devices by their serial numbers and attribute values that make sense to the specific location or function of that placement.

The following examples to illustrate the use of projects and placements.

Example 1:

In the `SalesPersonNotification` project, 10 customers receive a button they can press to contact a salesperson. There are 10 placements, one for each customer. Each placement has values for `CustomerName` (for example, Mr. Jones), `SalesPersonPhoneNumber` (for example, 1-555-555-1234), and button serial number (for example, G030PM12345678). The device template, `NotficationButton`, is contained in the placement. The `CustomerName` and `SalesPersonPhoneNumber` attributes are defined for each placement. When a customer clicks the button, AWS IoT 1-Click invokes `SendSMSLambda` with the `CustomerName` and `SalesPersonPhoneNumber` values associated with that button. The SMS is sent based on those values.

- **Placement template**:
  - Since each customer gets one button to notify a sales person, one device template named `NotficationButton` is created.
  - The device template (contained in the placement) specifies that the `NotificationButton`, when clicked, will call the `SendSMSLambda` Lambda function.
  - Attributes called `CustomerName` and `SalesPersonPhoneNumber` are defined for each placement.
- **Placements**: 10 placements are created, one per customer. Each placement has specific values for `CustomerName` (e.g., "Mr. Jones"), `SalesPersonPhoneNumber` (e.g., 1-555-555-1234) and button serial number (e.g., G030PM12345678).
- **Operation**: When a customer clicks his button, AWS IoT 1-Click invokes `SendSMSLambda` with the `CustomerName` and `SalesPersonPhoneNumber` values associated with that particular button - and an SMS is sent based on those values.

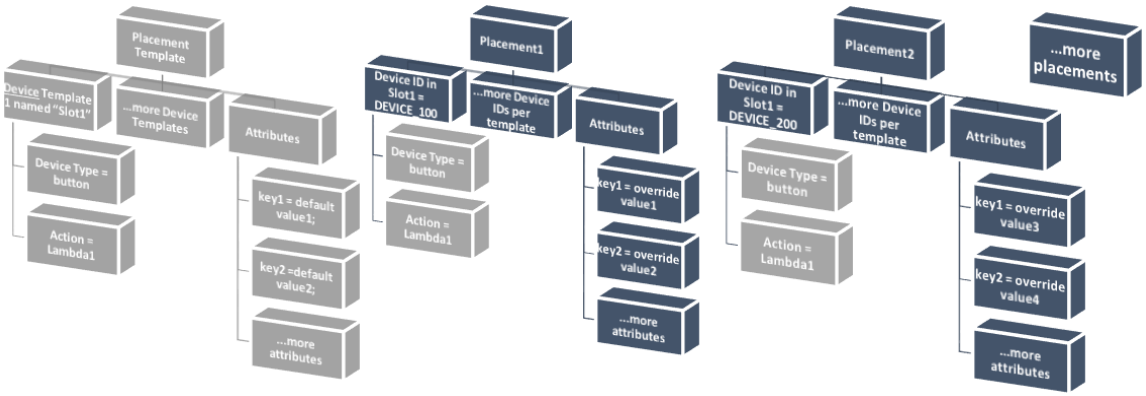
Example 2:

In the `MeetingRoomFeedback` project, user satisfaction is tracked through the pressing of Thumbs Up and Thumbs Down buttons in each of 50 conference rooms. There are two device templates, `ThumbsUp` and `ThumbsDown`.When the Thumbs Up button is clicked, the `PostiveFeedbackLambda` function

is called. When the Thumbs Down button is clicked, the `NegativeFeedbackLambda` is called. A `MeetingRoomNumber` attribute is defined to hold the room number for each placement. 50 device placements are created, one per conference room. Each placement contains the `MeetingRoomNumber` key set to a room number (for example, 1001) and two buttons, as identified by their unique serial numbers (for example, G030PM12345678 and G030PM23456789). When a button is clicked in a meeting room, AWS IoT 1-Click invokes the `PositiveFeedbackLambda` function or `NegativeFeedbackLambda` function with the `MeetingRoomNumber` value. Feedback can then be processed and tabulated.

- **Project name**: `MeetingRoomFeedback`
- **Placement template**:
  - Since each room gets two buttons, two device templates are created, respectively named `ThumbsUp` and `ThumbsDown`.
  - The device templates specify that `ThumbsUp` buttons will call `PostiveFeedbackLambda` when clicked, and that `ThumbsDown` buttons will call `NegativeFeedbackLambda` when clicked.
  - An attribute called `MeetingRoomNumber` is defined to hold the room number for each placement.
- **Placements**: 50 device placements are created, one placement per room. Each placement contains the `MeetingRoomNumber` key set to a particular room number pair (e.g., 1001) and two buttons as identified by their unique serial numbers (e.g., G030PM12345678 and G030PM23456789).
- **Operation**: When a button is clicked in a meeting room, AWS IoT 1-Click invokes the `PositiveFeedbackLambda` function or `NegativeFeedbackLambda` function with the `MeetingRoomNumber` value – and feedback can be processed and tabulated.

The following diagram shows these concepts:



For more information, see Getting Started with the AWS IoT 1-Click Console (p. 6).

Getting Started with the AWS IoT 1-Click Console

The following topics describe how to do common AWS IoT 1-Click tasks.

Topics

Claiming Devices

The following procedure shows you how to claim one or more AWS IoT 1-Click supported devices.

1. Sign in to your AWS account. If you do not have an AWS account, open https://aws.amazon.com/, choose **Create an AWS Account**, and follow the online instructions.
2. From the AWS Management Console, search for "1-Click," and then choose AWS IoT 1-Click.
3. If you're using one or more AWS IoT Enterprise Buttons, install the AWS IoT 1-Click mobile app for iOS or Android and connect the buttons to your local Wi-Fi network. The AWS IoT 1-Click mobile app is available from the **Onboard** page of the AWS IoT 1-Click console. This step is not required for the LTE-M Button because it uses the cellular network.
4. Choose **Onboard**, and then choose **Claim devices**.
5. Enter one or more device IDs (p. 1) (such as a device serial number) or claim codes (p. 1), separated by commas, and then choose **Claim**. If the **Claim** button is unavailable, double-check all the values you entered.
6. Press the button(s) on your device(s), and then choose **Done**. A list of all known devices should be displayed.

Creating a Project

The following procedure shows you how to create an AWS IoT 1-Click project for your AWS IoT 1-Click supported device(s).

1. Sign in to your AWS account and open the AWS IoT 1-Click console.
2. Choose **Onboard**, and then choose **Create a project**.
3. Type a name and optional description for the project, and then choose **Next**.
4. To define one or more templates for your placement, under **Program a device template**, choose **Start**.
5. To define a template for any button device, choose **All button types**.
6. For **Device template name**, type a descriptive name for your template. Under **Action**, choose **Send SMS** or **Send email**. You can use the **Custom action using a Lambda function** option and chooseone of your own Lambda functions. Enter the phone number, email address, or Lambda function name, depending on your choice. For more information about creating Lambda functions, see the AWS Lambda Developer Guide.
7. Under **Add another device template (if you need multiple devices per placement)** choose **Add**.

8. Enter an attribute key-value pair. You can enter additional key-value pairs, if necessary.
9. Choose **Create project**.

The following section, Example: Meeting Room Satisfaction Project (p. 7), provides a real-world example of how to use the AWS IoT 1-Click console to create a project.

Example: Meeting Room Satisfaction Project

The following example might help you understand AWS IoT 1-Click concepts.

- A project to track the satisfaction of 50 meeting rooms (and associated AV equipment) is created and named `MeetingRoomSat`.
- Each meeting room will receive two devices (buttons), one physically marked "Satisfied" and the other marked "Unsatisfied". Since there are two buttons per room, two templates are created, one named `Satisfied` and the other named `Unsatisfied`.
- The `Satisfied` template is configured to invoke a Lambda function called `SatLambda`.
- The `Unsatisfied` template is configured to invoke a Lambda function called `UnsatLambda`.
- For both of these templates, an attribute (key/value pair) named `MeetingRoomNum` (key) is created whose value is `TBD` (the `TBD` value will be changed to the room number when both buttons are physically placed in a room).
- 50 placements are created, one for each room. Each placement has the two templates associated with it (i.e., `Satisfied` and `Unsatisfied`).
- Two buttons are physical labeled and placed in a room. Then, using the AWS IoT 1-Click mobile app or the AWS IoT 1-Click console and the button's serial numbers, the "Satisfied" and "Unsatisfied" marked buttons are associated with one of the 50 placements. This process continues until all remaining placements are deployed.
- When a room button is clicked in a meeting room, AWS IoT 1-Click invokes either the `SatLambda` or `UnsatLambda` function with the `MeetingRoomNum` value – and feedback can be processed and stored in the cloud.
- Later, another template can be added to the project so that the 50 existing placements now contain a slot for a new button to indicate that more towels or other toiletries are needed in each bathroom.

The following provides an example of using the AWS IoT 1-Click console to create a project for monitoring meeting room satisfication in an office building (as part of a group of office buildings).

To monitor the satisfaction of meeting rooms, including their audio/video equipment, two AWS IoT Enterprise buttons, one labeled "Satisfied" and the other labeled "Unsatisifed", are placed in each meeting room. This is a pilot project, the results of which can be used to improve meeting room customer satisfaction in other buildings on campus.

At the end of a meeting, participants are encouraged to press either the "Satisfied" or "Unsatisfied" button to record their overall satisfaction with the meeting room and its equipment. This data is then used to identify meeting rooms with non-functional A/V equipment or other problems.

The AWS IoT 1-Click console can be used to set up this project:

1. From the AWS IoT 1-Click console, choose **Create a project**.
2. For the project name, type **`MeetingRoomSatisfaction`**. For the project description, type **`Project used to track customer meeting room satisfaction, including A/V equipment.`** Choose **Next**.
3. Under **Program a device template** choose **Start**, and then choose **All button types**.
4. For **Device template name**, type **`Satisfied`**. This is the template used for all buttons labeled "Satisfied." For **Action**, choose **Send email**.

Note
If the meeting room satisfication pilot is successful, under **Action**, you might choose **Custom action using a Lambda function**. This custom Lambda function could send an email or store the "Satisfied" button data in an Amazon DynamoDB table for later analysis. For information about creating Lambda functions, see the AWS Lambda Developer Guide.

5. Under **Add another device template (if you need multiple devices per placement)** choose **Add**, and then choose **All button types**. For **Device template name**, type **`Unsatisfied`**. This is the template used for all buttons labeled "Unsatisfied." For **Action**, choose **Send email**.
6. For **Required email default value**, type an email address. For **Required subject default value**, type **`Meeting Room Feedback`**. For **Required body default value**, type **`Either positive or negative meeting room feedback has been provided.`**
7. For **Attribute key**, type **`Building`**. For **Default value**, type **`Headquarters`**. The meeting room satisfaction pilot is taking place in the company's headquarters building. If the pilot is successful, it will be deployed to the company's other buildings. Therefore, it's important to know from which building the meeting room devices are providing information for.
8. In the second key-value pair row, for **Attribute key**, type **`Room`**. For **Default value**, type **`TBD`**. The **`TBD`** value will be changed to a meeting room number when the buttons are placed there (using either the AWS IoT 1-Click mobile app or AWS IoT 1-Click console).
9. Choose **Create project**.

Using the AWS IoT 1-Click mobile app, when a "Satisfied" button is placed in a meeting room, the **Satisfied** template becomes associated with it and the **`TBD`** value is replaced with the meeting room number. The same is true for when the "Unsatisfied" button is placed in a meeting room.

AWS IoT 1-Click Mobile App

The AWS IoT 1-Click mobile app lets you:

- Conveniently configure and monitor AWS IoT 1-Click devices in the field using a user interface similar to the AWS IoT 1-Click console.
- Configure Wi-Fi credentials for Wi-Fi-connected AWS IoT 1-Click devices (such as the AWS IoT Enterprise Button).

The AWS IoT 1-Click mobile app is available for iPhone and Android mobile devices. To download the app, go to the App Store or Google Play, and search for AWS IoT 1-Click.

AWS IoT 1-Click Programming Model

To build applications using AWS IoT 1-Click devices, programmers use the AWS IoT 1-Click Devices API and the AWS IoT 1-Click Projects API. The Devices API interacts with the AWS IoT 1-Click devices component and handles events coming from devices. These events include enabling and disabling the devices and defining event formats and the actions (Lambda functions) that they trigger. The Devices API is tightly coupled with the AWS components that reside in the region where the manufacturer registered the devices. This is why AWS device regions (p. 3) might be different from the region where the customer is using the devices. The Projects API interacts with the AWS IoT 1-Click Projects service and is used to manage AWS IoT 1-Click devices in aggregate, which makes it possible to:

- Group devices into projects.
- Create templates used to set actions for all devices in the project.
- Define attributes storing contextual data pertinent to the project.

You can use the AWS IoT 1-Click programming model to program individual devices by using the Devices API. In this case, you will use the AWS IoT 1-Click device type. The API defines standard event formats and a list of methods that form the programming interface for all devices of that type. To invoke methods pertaining to a given device type, a programmer can use the `InvokeDeviceMethod` API and specify the device method as a parameter.

For example, all AWS IoT 1-Click devices that have the device type "button" emit events associated with clicks and have methods to set callback functions that are invoked when the device is clicked. For information about the button interface, see Interfaces by Device Type (p. 3). Here is the code to set this callback function:

```
    String methodParameters = mapper.writeValueAsString(
        SetOnClickCallbackRequestParameters.builder()
                .deviceId(deviceId)
                .callback(DeviceCallback.builder()
                        .awsLambdaArn(“arn:aws:lambda:us-
west-2:123456789012:MyButtonListener”)
                        .build())
                .build());
    InvokeDeviceMethodRequest request = new InvokeDeviceMethodRequest()
        .withDeviceMethod(new DeviceMethod()
                .withDeviceType("button")
                .withMethodName("setOnClickCallback"))
        .withDeviceMethodParameters(methodParameters);
```

You use the Projects API to program a fleet of devices. Using the APIs, you first define what each placement looks like, including device templates and attributes for each placement. After that is done, you create placements with specific device IDs. Each placement follows the same template. Here is the sample code to do this:

```
    final Map<String, String> callbacks = new HashMap<>();
callbacks.put("onClickCallback", "arn:aws:lambda:us-west-2:123456789012:MyButtonListener");
final DeviceTemplate item = DeviceTemplate.builder()
                .withDeviceType(“button”)
                .withCallbackOverrides(callbacks)
```

```
                .build();
final Map<String, DeviceTemplate> deviceTemplateMap = new HashMap<>();
deviceTemplateMap.put("MyDevice", item);

final Map<String, String> placementDefaultAttributes = new HashMap<>();
placementDefaultAttributes.put("location", "Seattle")

request = CreateProjectRequest.builder()
                .withProjectName("HelloWorld")
                .withDescription("My first project!")
                .withPlacementTemplate(PlacementTemplate.builder()
                        .withDefaultAttributes(placementDefaultAttributes)
                        .withDeviceTemplates(deviceTemplateMap)
                        .build())
                .build();
projectsClient.createProject(request)
```

AWS IoT 1-Click Callback Events

AWS IoT 1-Click allows you to subscribe to device events by registering callbacks. An example of a callback is an AWS Lambda function owned and implemented by you, the AWS IoT 1-Click customer. This callback is invoked every time there is an event available for it to consume. For information about events and their payloads, see the AWS IoT 1-Click Click Events (p. 11) and AWS IoT 1-Click Health Events (p. 12) sections.

AWS IoT 1-Click Click Events

Devices of type `button` publish a click event each time they are clicked. You can subscribe to this event by:

- Calling the device `SetOnClickCallback` method on a device.
- Configuring the associated project appropriately, as shown in the earlier create project code example.

In the following example, be aware that the `placementInfo` section is only present when the device has an associated placement. For more information, see Projects, Templates, and Placements (p. 4).

```
{
    "deviceEvent": {
      "buttonClicked": {
        "clickType": "SINGLE",
        "reportedTime": "2018-05-04T23:26:33.747Z"
      }
    },
    "deviceInfo": {
      "attributes": {
        "key3": "value3",
        "key1": "value1",
        "key4": "value4"
      },
      "type": "button",
      "deviceId": " G030PMXXXXXXXXXX ",
      "remainingLife": 5.00
    },
    "placementInfo": {
      "projectName": "test",
      "placementName": "myPlacement",
      "attributes": {
```

```
        "location": "Seattle",
        "equipment": "printer"
      },
      "devices": {
        "myButton": " G030PMXXXXXXXXXX "
      }
    }
  }
```

AWS IoT 1-Click Health Events

Devices publish a health event based on the health parameters calculated by the AWS IoT 1-Click service, but you set their corresponding thresholds. The following example represents the JSON payload of a health event for device `G030PMXXXXXXXXXX` with a remaining life of 10% (note `"remainingLifeLowerThan":10` key-value pair).

```
{
    "deviceEvent": {
      "deviceHealthMonitor": {
        "condition": {
          "remainingLifeLowerThan": 10
        }
      }
    },
    "deviceInfo": {
      "attributes": {
        "key2": "value2",
        "key1": "value1",
        "projectRegion": "us-west-2"
      },
      "type": "button",
      "deviceId": "G030PMXXXXXXXXXX",
      "remainingLife": 5.4
    }
  }
```

Device Methods

AWS IoT 1-Click device methods are APIs that are supported by devices of a certain device-type, as demonstrated in the following table. The full list of device methods supported by any device can be retrieved by calling `GetDeviceMethods`.

Device Type	Method Name	Description
`device`	`getDeviceHealthParameters`	Gets the device's health parameters, such as `remainingLife`.
`device`	`setDeviceHealthMonitorCallback`	Sets a callback to be called when the device health parameters are below a threshold.
`device`	`getDeviceHealthMonitorCallback`	Gets the configured callback which is called when the health parameters are below a threshold.

Device Type	Method Name	Description
`button`	`setOnClickCallback`	Sets a callback to be called when the button has been clicked.
`button`	`getOnClickCallback`	Gets the configured callback which is called when the button is clicked.

Monitoring AWS IoT 1-Click with Amazon CloudWatch

AWS IoT 1-Click automatically monitors devices on your behalf and reports metrics through Amazon CloudWatch. These metrics are reported in the device region where the devices were registered by the manufacturer. For more information about device regions, see How AWS IoT 1-Click Works (p. 3). You can find the metrics in the Amazon CloudWatch dashboard under the **IoT1Click** namespace.

Amazon CloudWatch Events enables you to automate your AWS services and respond automatically to system events, such as application availability issues or resource changes. Events from AWS services are delivered to CloudWatch Events in near real time. You can write simple rules to indicate which events are of interest to you and which automated actions to take when an event matches a rule. The following actions that can be triggered:

- Invoking an AWS Lambda function.
- Invoking Amazon EC2 Run Command.
- Relaying the event to Amazon Kinesis Data Streams.
- Activating an AWS Step Functions state machine.
- Notifying an Amazon SNS topic or an AWS SMS queue.

AWS IoT 1-Click tracks and reports the following metrics:

- **TotalEvents** tracks the number of events published by devices. This metric can be viewed and graphed by device event, project, device type, or product type.
- **RemainingLife** represents the approximate percentage of life remaining for a device. AWS IoT 1-Click reports this number based on the manufacturer's rating of the device. For example, if a button is designed to last for approximately 2000 clicks, and 500 clicks have been recorded, the **RemainingLife** value is reported as 75%. The **RemainingLife** metric can be viewed and graphed by project, device type, or product type. Customers can use the **RemainingLife** metric to set up alarms that are triggered when devices fall below a certain threshold. Customers can then query the **RemainingLife** of devices by using the `GetDeviceHealthParameters` method to identify devices that have low **RemainingLife** values.
- **CallbackInvocationErrors** tracks failures in invoking the callbacks (Lambda functions) when the device emits an event. The **CallbackInvocationErrors** metric can be viewed and graphed by invoked callback (Lambda function ARNs set as callbacks) or by project. Customers can set up alarms for the **CallbackInvocationErrors** metric to be notified when AWS IoT 1-Click was unable to route events from their devices to their configured Lambda functions.

For more information, see the Amazon CloudWatch Events User Guide.

Logging AWS IoT 1-Click API Calls with AWS CloudTrail

AWS IoT 1-Click 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 1-Click. CloudTrail captures API calls for AWS IoT 1-Click as events. The calls captured include calls from the AWS IoT 1-Click console and code calls to the AWS IoT 1-Click API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS IoT 1-Click. 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 1-Click, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, including how to configure and enable it, see the AWS CloudTrail User Guide.

AWS IoT 1-Click Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in AWS IoT 1-Click, 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 1-Click, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

The AWS IoT 1-Click Devices API supports logging the following actions as events in CloudTrail log files:

- ListDevices
- DescribeDevice
- GetDeviceMethods
- UpdateDeviceState
- InvokeDeviceMethod

The AWS IoT 1-Click Projects API supports logging the following actions as events in CloudTrail log files:

- CreateProject
- UpdateProject

- DescribeProject
- ListProjects
- DeleteProject
- CreatePlacement
- UpdatePlacement
- DescribePlacement
- ListPlacements
- DeletePlacement
- AssociateDeviceWithPlacement
- DisassociateDeviceFromPlacement
- GetDevicesInPlacement

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.

Example: AWS IoT 1-Click 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 aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `DescribeDevice` action.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::012345678910:user/Alice",
        "accountId": "012345678910",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2018-04-12T18:57:27Z",
    "eventSource": "iot1click.amazonaws.com",
    "eventName": "DescribeDevice",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "console.aws.amazon.com",
    "requestParameters": {
        "deviceId": "G030PM12345678"
    },
    "responseElements": null,
    "requestID": "573c5654-3e83-11e8-9eac-c999bd01134e",
    "eventID": "be323b62-082a-4352-929d-085d2a3249b0",
```

```
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "012345678910"
}
```

The following example shows a CloudTrail log entry that demonstrates the `CreateProject` action.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::012345678910:user/Alice",
        "accountId": "012345678910",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2018-04-12T20:31:02Z",
    "eventSource": "iot1click.amazonaws.com",
    "eventName": "CreateProject",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "console.aws.amazon.com",
    "requestParameters": {
        "description": "",
        "placementTemplate": {
            "defaultAttributes": "***",
            "deviceTemplates": {
                "happyId": {
                    "deviceType": "button",
                    "callbackOverrides": {
                        "onClickCallback": "arn:aws:lambda:us-
west-2:012345678910:function:rating_buttons_happy"
                    }
                },
                "sadId": {
                    "deviceType": "button",
                    "callbackOverrides": {
                        "onClickCallback": "arn:aws:lambda:us-
west-2:012345678910:function:rating_buttons_sad"
                    }
                }
            }
        }
    }
}
```

AWS CloudFormation Integration

AWS IoT 1-Click is integrated with AWS CloudFormation, a common language for you to describe and provision all your infrastructure resources in a cloud environment (i.e., Amazon EC2, Auto Scaling, Amazon SNS, etc.) AWS CloudFormation allows you to use a simple text file to model and provision, in an automated and secure manner, all the resources needed for your applications across all regions and accounts. This file serves as the single source of truth for your cloud environment. For more information, see the AWS CloudFormation User Guide as well as the AWS IoT 1-Click topics (such as AWS::IoT1Click::Project) in the *AWS CloudFormation User Guide*.

Authentication and Access control for AWS IoT 1-Click

Access to AWS IoT 1-Click APIs requires credentials. Those credentials must have permissions to access AWS resources, such as an AWS IoT 1-Click project or device. The following sections provide details on how you can use AWS Identity and Access Management (IAM) and AWS IoT 1-Click to help secure access to your resources.

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles), and some services (such as AWS Lambda) also support attaching permissions policies to resources. When granting permissions, the administrator decides who is getting the permissions, the resources they get permissions for, and the specific actions that they want to allow on those resources.

AWS IoT 1-Click Resources and Operations

In AWS IoT 1-Click, the primary resources are projects and devices. In a policy, you use an Amazon Resource Name (ARN) to identify the resource that the policy applies to. These resources have unique Amazon Resource Names (ARNs) associated with them, as shown in the following table.

Resource Type	ARN Format
Device	arn:aws:iot1click:region:account-id:devices/device-id
Project	arn:aws:iot1click:region:account-id:projects/project-name

AWS IoT 1-Click implements APIs to work with AWS IoT 1-Click resources. These are referred to as Actions in IAM. For a list of available operations, see the table at the end of this topic.

Using Identity-Based Policies (IAM Policies) for AWS IoT 1-Click

This topic provides examples of identity-based policies that demonstrate how an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles) and thereby grant permissions to perform operations on AWS IoT 1-Click resources.

The following shows an example of a permissions policy.

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

```
            "Effect": "Allow",
            "Action": [
                "iot1click:CreateProject"
            ],
            "Resource": "*"
        }
    ]
}
```

The policy has one statement - which grants permissions for one AWS IoT 1-Click action (`iot1click:CreateProject`) on a resource using the Amazon Resource Name (ARN) for the application. The ARN in this case specifies a wildcard character (*) to indicate that the permission is granted for any resource.

For a table showing all of the AWS IoT 1-Click API operations and the resources that they apply to, see AWS IoT 1-Click API Permissions: Actions, Permissions, and Resources Reference (p. 20).

AWS Managed (Predefined) Policies for AWS IoT 1-Click

Amazon Web Services addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These AWS managed policies grant necessary permissions for common use cases so that you can avoid having to investigate what permissions are needed. For more information, see AWS Managed Policies in the *IAM User Guide*.

The following AWS managed policies, which you can attach to users in your account, are specific to AWS IoT 1-Click and are grouped by use case scenario:

- `AWSIoT1ClickFullAccess`: Grants full access to AWS IoT 1-Click resources by using the AWS Management Console. The granted permissions include all AWS IoT 1-Click actions to manage devices and projects.
- `AWSIoT1ClickReadOnlyAccess`: Grants read-only access to AWS IoT 1-Click resources by using the AWS Management Console. This access enables a user to list AWS IoT 1-Click devices and projects, and to review project configuration.

> **Note**
> You can review these permission policies by signing in to the IAM Console (https://console.aws.amazon.com/iam/) and searching for the specific policy name(s) there.

You can also create your own custom IAM policies to allow permissions for AWS IoT 1-Click actions and resources. You can attach these custom policies to the IAM users or groups that require those permissions.

AWS IoT 1-Click API Permissions: Actions, Permissions, and Resources Reference

When you are setting up Access Control in the AWS Cloud and writing a permissions policy that you can attach to an IAM identity (identity-based policies), you can use the following table as a reference. The table lists each AWS IoT 1-Click API operation, the corresponding actions for which you can grant permissions to perform the action, and the AWS resource for which you can grant the permissions. You specify the actions in the policy's `Action` field, and you specify the resource value in the policy's `Resource` field.

You can use AWS-wide condition keys in your AWS IoT 1-Click policies to express conditions. For a complete list of AWS-wide keys, see Available Keys in the *IAM User Guide*.

Note
To specify an action, use the `iot1click:` prefix followed by the API operation name (for example, `iot1click:ListProjects`).

IoT 1-Click Operations	Required Permissions (API Actions)	Resources
ListDevices	iot1click:ListDevices	*
DescribeDevice	iot1click:DescribeDevice	arn:aws:iot1click:region:account-id:devices/device-id
GetDeviceMethods	iot1click:GetDeviceMethods	arn:aws:iot1click:region:account-id:devices/device-id
UpdateDeviceState	iot1click:UpdateDeviceState	arn:aws:iot1click:region:account-id:devices/device-id
InvokeDeviceMethod	iot1click:InvokeDeviceMethod	arn:aws:iot1click:region:account-id:devices/device-id
ListDeviceEvents	iot1click:ListDeviceEvents	arn:aws:iot1click:region:account-id:devices/device-id
InitializeDeviceClaim	iot1click:InitializeDeviceClaim	arn:aws:iot1click:region:account-id:devices/device-id
FinalizeDeviceClaim	iot1click:FinalizeDeviceClaim	arn:aws:iot1click:region:account-id:devices/device-id
UnclaimDevice	iot1click:UnclaimDevice	arn:aws:iot1click:region:account-id:devices/device-id
ClaimDeviceByClaimCode	iot1click:ClaimDeviceByClaimCode	*
CreateProject	iot1click:CreateProject	arn:aws:iot1click:region:account-id:projects/project-name
UpdateProject	iot1click:UpdateProject	arn:aws:iot1click:region:account-id:projects/project-name
DescribeProject	iot1click:DescribeProject	arn:aws:iot1click:region:account-id:projects/project-name
ListProjects	iot1click:ListProjects	*
DeleteProject	iot1click:DeleteProject	arn:aws:iot1click:region:account-id:projects/project-name
CreatePlacement	iot1click:CreatePlacement	arn:aws:iot1click:region:account-id:projects/project-name
UpdatePlacement	iot1click:UpdatePlacement	arn:aws:iot1click:region:account-id:projects/project-name
DescribePlacement	iot1click:DescribePlacement	arn:aws:iot1click:region:account-id:projects/project-name
ListPlacements	iot1click:ListPlacements	arn:aws:iot1click:region:account-id:projects/project-name

IoT 1-Click Operations	Required Permissions (API Actions)	Resources
DeletePlacement	iot1click:DeletePlacement	arn:aws:iot1click:region:account-id:projects/project-name
AssociateDeviceWithPlacement	iot1click:AssociateDeviceWithPlacement	arn:aws:iot1click:region:account-id:projects/project-name
DissacociateDeviceFromPlacement	iot1click:DissacociateDeviceFromPlacement	arn:aws:iot1click:region:account-id:projects/project-name
GetDevicesInPlacement	iot1click:GetDevicesInPlacement	arn:aws:iot1click:region:account-id:projects/project-name

Tagging Your AWS IoT 1-Click Resources

To help you manage your AWS IoT 1-Click resources, you can optionally assign your own metadata to any ARN-based resource using tags. This chapter describes tags and shows you how to create them.

Tag Basics

Tags enable you to categorize your AWS IoT 1-Click 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 search for and identify a specific 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 could define a set of tags for a multiple buttons owned by a particular manager or account. You can search and filter the resources based on the tags you add. We recommend that you devise a set of tag keys that meets your needs for each resource type. Using a consistent set of tag keys makes it easier for you to manage your resources. For more information, see AWS Tagging Strategies.

You can also use tags to categorize and track your costs. When you apply tags to resources, 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.

For ease of use, you can use the Tag Editor in the AWS Management Console, which provides a central, unified way to create and manage your tags. For more information, see Working with Tag Editor in Getting Started with the AWS Management Console.

You can also work with tags using the AWS CLI and the AWS IoT 1-Click Device and Project APIs. You can associate tags with AWS IoT 1-Click projects and devices when you create them by using the *tags* field in the following commands:

- CreateProject (Projects API)
- Finalize Claim (Devices API)

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

AWS IoT 1-Click Projects API (using project ARNs)	**AWS IoT 1-Click Devices API (using device ARNs)**
TagResource	Tag See `TagResource` (POST), `TagListTagsForResource` (GET), and `UntagResource` (DELETE).
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

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 because it is reserved for AWS use. You can't edit or delete tag names or values with this prefix. Tags with this prefix do not count against your tags per resource limit.
- If your tagging schema is used across multiple services and resources, remember that other services may have restrictions on allowed characters. Generally, allowed characters are: letters, spaces, and numbers representable in UTF-8, plus the following special characters: + - = . _ : / @

AWS IoT Enterprise Button User Guide

The AWS IoT Enterprise Button is a simple and easy to configure Wi-Fi-based button. It is designed for enterprises and developers to easily integrate with existing business workflows and systems using AWS IoT 1-Click.

The AWS IoT Enterprise Button supports three types of clicks:

- Single
- Double
- Long press

To function correctly, you must configure the button's Wi-Fi connection using the AWS IoT 1-Click mobile app (iOS or Android). In the app, you can configure the button's Wi-Fi connection by logging in to your AWS account, or by tapping the Wi-Fi icon in the upper-right of the app to skip logging in.

After the connection is configured and claimed through the mobile app or console, the button should flash solid green when a single, double, or long press click occurs.

If you suspect there is an issue with the button after configuring it, this table can help you troubleshoot.

Color	Status	Recommendation
Blinking white	Connecting to Wi-Fi, obtaining IP address, or connecting to AWS IoT.	N/A
Solid green	Successfully connected to Wi-Fi and published a message to AWS IoT.	N/A
Blinking blue	Button is in configuration mode.	Wait for the configuration process to complete.
Solid orange	Wi-Fi not configured.	Use the AWS IoT 1-Click mobile app to configure Wi-Fi.
Red: short, short, short	There was an error connecting to the configured wireless network.	Check if any network settings have changed or if the button is too far away from the Wi-Fi router.
Red: short, short, long	There was an error obtaining an IP address from the wireless network.	Check for wireless network issues.
Red: short, long, short	There was an error performing a host name lookup.	Check for wireless network issues.

Color	Status	Recommendation
Red: short, long, long	Cannot connect to AWS IoT.	Check for wireless network issues. If no network issues exist and the problem persists, contact AWS Support Center and provide the device serial number (DSN). You'll find it on the back of the button.
Red: long, short, short	Cannot establish a secure connection with the server.	Use the AWS IoT 1-Click iOS or Android mobile apps to check if you have the latest firmware.
Red: long, short, long	Received an HTTP 403 forbidden error.	Contact the AWS Support Center and provide the DSN. You'll find it on the back of the button.
Red: after 15 second button press	Button reset.	You can reset the AWS IoT Enterprise Button Wi-Fi configuration by pressing the button for 15 seconds.

Using AWS IoT 1-Click with the AWS CLI

To demonstrate the use of the AWS Command Line Interface (AWS CLI), consider the scenario of a disposal company that wants to streamline its dumpster pickup service using AWS IoT 1-Click.

In this scenario, each dumpster is paired with an AWS IoT Enterprise Button. When a dumpster is full, the customer only needs to press the associated button to request that the dumpster be replaced.

Note
All AWS IoT Enterprise Button device IDs start with "G030PM".

The following steps are used by the garbage disposal company to prepare the AWS IoT Enterprise Button for customer use.

To prepare the AWS IoT Enterprise Button for customer use

1. The only way to set up Wi-Fi for an AWS IoT Enterprise Button is to use the AWS IoT 1-Click mobile app. To install the app, see AWS IoT 1-Click Mobile App (p. 9). After you install the app, don't press **Login to AWS Account** (as would normally be the case). In this exercise, we want to demonstrate how to use the AWS CLI. If you press **Login to AWS Account**, the **initiate-device-claim** and **finalize-device-claim** commands are invoked for you, and we want to "manually" do this using the CLI as shown in the following steps.
2. For AWS CLI demonstration purposes, instead of pressing **Login to AWS Account**, choose the small round Wi-Fi icon in the upper-right corner. Next, choose **Configure Wi-Fi**. Scan or enter the device ID and follow the rest of the mobile app's instructions.
3. If you don't have the AWS CLI installed, follow the instructions in Installing the AWS CLI. To list the available AWS IoT 1-Click AWS CLI commands, run the following two commands.

```
aws iot1click-projects help
```

```
aws iot1click-devices help
```

4. To associate the now Wi-Fi connected AWS IoT Enterprise Button to the garbage disposal company's AWS account, run the following command using the device ID from your device.

```
aws iot1click-devices initiate-device-claim --device-id G030PM0123456789
{
    "State": "CLAIM_INITIATED"
}
```

   Press the button on the device. After some intermittent flashing of white light, you should see solid green light for about one second. If you don't, repeat the previous Wi-Fi connection procedure.
5. After you see the solid green light in the previous step, run the following command (using your device's ID value).

```
aws iot1click-devices finalize-device-claim --device-id G030PM0123456789
{
    "State": "CLAIMED"
}
```

   The `"State": "CLAIMED"` response indicates that the device is successfully registered with the AWS IoT 1-Click service.

Note
If the device manufacturer provided a claim code starting with "C-", you can use just the **aws iot1click-devices claim-devices-by-claim-code** command to claim one or more devices using a single claim code, as shown in the following example.

```
aws iot1click-devices claim-devices-by-claim-code --claim-code C-123EXAMPLE
{
    "Total": 9
    "ClaimCode": "C-123EXAMPLE"
}
```

In this example, `"Total": 9` indicates that the nine devices associated with claim code `C-123EXAMPLE` have been successfully claimed by the AWS IoT 1-Click service.

6. Next, you prepare to create an appropriate AWS IoT 1-Click project for the disposal company by creating a JSON text file named `create-project.json`. This file contains the following.

```
{
   "projectName": "SeattleDumpsters",
   "description": "All dumpsters in the Seattle region.",
   "placementTemplate": {
      "defaultAttributes": {
         "City" : "Seattle"
      },
      "deviceTemplates": {
         "empty-dumpster-request" : {
            "deviceType": "button"
         }
      }
   }
}
```

The `placementTemplate` and `deviceTemplates` key-value pairs are the attributes that will be applied to all buttons that are part of the `SeattleDumpsters` project. To create this project, run the following command (which assumes `create-project.json` is in the current working directory of the AWS CLI command prompt).

```
aws iot1click-projects create-project --cli-input-json file://create-project.json
```

To view the newly created project, run the following command.

```
aws iot1click-projects list-projects
{
    "projects": [
        {
            "arn": "arn:aws:iot1click:us-west-2:012345678901:projects/
SeattleDumpsters",
            "projectName": "SeattleDumpsters",
            "createdDate": 1563483100,
            "updatedDate": 1563483100,
            "tags": {}
        }
    ]
}
```

For greater detail, run the **describe-project** command, as follows.

```
aws iot1click-projects describe-project --project-name SeattleDumpsters
{
```

```
    "project": {
        "arn": "arn:aws:iot1click:us-west-2:012345678901:projects/SeattleDumpsters",
        "projectName": "SeattleDumpsters",
        "description": "All dumpsters in the Seattle region.",
        "createdDate": 1563483100,
        "updatedDate": 1563483100,
        "placementTemplate": {
            "defaultAttributes": {
                "City": "Seattle"
            },
            "deviceTemplates": {
                "empty-dumpster-request": {
                    "deviceType": "button",
                    "callbackOverrides": {}
                }
            }
        },
        "tags": {}
    }
}
```

7. With the project created for the Seattle region, next you create a placement for a particular dumpster (for customer 217), as follows. The escaped quotation marks are necessary for Windows.

```
aws iot1click-projects create-placement --project-name SeattleDumpsters --placement-
name customer217 --attributes "{\"location\": \"1800 9th Ave Seattle, WA 98101\",
 \"phone\": \"206-123-4567\"}"
```

   To view the newly created placement, run the following command.

```
aws iot1click-projects list-placements --project-name SeattleDumpsters
{
    "placements": [
        {
            "projectName": "SeattleDumpsters",
            "placementName": "customer217",
            "createdDate": 1563488454,
            "updatedDate": 1563488454
        }
    ]
}
```

   For greater detail, run the **describe-placement** command, as follows.

```
aws iot1click-projects describe-placement --project-name SeattleDumpsters --placement-
name customer217
{
    "placement": {
        "projectName": "SeattleDumpsters",
        "placementName": "customer217",
        "attributes": {
            "phone": "206-123-4567",
            "location": "1800 9th Ave Seattle, WA 98101"
        },
        "createdDate": 1563488454,
        "updatedDate": 1563488454
    }
}
```

8. Although the device is now associated with the disposal company's AWS IoT 1-Click account, it's not associated with the placement. Confirm this by running the following command.

```
aws iot1click-projects get-devices-in-placement --project-name SeattleDumpsters --
placement-name customer217
{
    "devices": {}
}
```

To associate the device with the placement, run the following command.

```
aws iot1click-projects associate-device-with-placement --project-name SeattleDumpsters
 --placement-name customer217 --device-template-name empty-dumpster-request --device-id
 G030PM0123456789
```

To confirm the previous command, run **get-devices-in-placement** again.

```
aws iot1click-projects get-devices-in-placement --project-name SeattleDumpsters --
placement-name customer217
{
    "devices": {
        "empty-dumpster-request": "G030PM0123456789"
    }
}
```

For greater detail, run the **describe-device** command, as follows (notice the switch from **iot1click-projects** to **iot1click-devices**).

```
aws iot1click-devices describe-device --device-id G030PM0123456789
{
    "DeviceDescription": {
        "Arn": "arn:aws:iot1click:us-west-2:012345678901:devices/G030PM0123456789",
        "Attributes": {
            "projectRegion": "us-west-2",
            "projectName": "SeattleDumpsters",
            "placementName": "customer217",
            "deviceTemplateName": "empty-dumpster-request"
        },
        "DeviceId": "G030PM0123456789",
        "Enabled": false,
        "RemainingLife": 99.9,
        "Type": "button",
        "Tags": {}
    }
}
```

Because there's currently only one device, the following command produces similar results.

```
aws iot1click-devices list-devices --device-type button
{
    "Devices": [
        {
            "Arn": "arn:aws:iot1click:us-west-2:012345678901:devices/G030PM0123456789",
            "Attributes": {
                "projectRegion": "us-west-2",
                "projectName": "SeattleDumpsters",
                "placementName": "customer217",
                "deviceTemplateName": "empty-dumpster-request"
            },
            "DeviceId": "G030PM0123456789",
            "Enabled": false,
            "RemainingLife": 99.9,
```

```
            "Type": "button",
            "Tags": {}
        }
    ]
}
```

9. To verify that the device is functioning properly, run the following command. Adjust the timestamps appropriately, which are in ISO 8061 format.

```
aws iot1click-devices list-device-events --device-id G030PM0123456789 --from-time-stamp
 2019-07-17T15:45:12.880Z --to-time-stamp 2019-07-19T15:45:12.880Z
{
    "Events": [
        {
            "Device": {
                "Attributes": {},
                "DeviceId": "G030PM0123456789",
                "Type": "button"
            },
            "StdEvent": "{\"clickType\": \"SINGLE\",
 \"reportedTime\": \"2019-07-18T23:47:55.015Z\", \"certificateId\":
 \"fe8798a6c97c62ef8756b80eeefdcf2280f3352f82faa8080c74cc4f4a4d1811\", \"remainingLife
\": 99.85000000000001, \"testMode\": false}"
        }
    ]
}
```

Here we see that a single-click event (`\"clickType\": \"SINGLE\"`) occurred on 2019-07-18T23:47:55.015Z. Now double-click the device (two quick button presses in succession), and run the command again. Now notice the double-click event (`\"clickType\": \"DOUBLE\"`), similar to the following.

```
aws iot1click-devices list-device-events --device-id G030PM0123456789 --from-time-stamp
 2019-07-17T15:45:12.880Z --to-time-stamp 2019-07-19T15:45:12.880Z
{
    "Events": [
        {
            "Device": {
                "Attributes": {},
                "DeviceId": "G030PM0123456789",
                "Type": "button"
            },
            "StdEvent": "{\"clickType\": \"SINGLE\",
 \"reportedTime\": \"2019-07-18T23:47:55.015Z\", \"certificateId\":
 \"fe8798a6c97c62ef8756b80eeefdcf2280f3352f82faa8080c74cc4f4a4d1811\", \"remainingLife
\": 99.85000000000001, \"testMode\": false}"
        },
        {
            "Device": {
                "Attributes": {},
                "DeviceId": "G030PM0123456789",
                "Type": "button"
            },
            "StdEvent": "{\"clickType\": \"DOUBLE\",
 \"reportedTime\": \"2019-07-19T00:14:41.353Z\", \"certificateId\":
 \"fe8798a6c97c62ef8756b80eeefdcf2280f3352f82faa8080c74cc4f4a4d1811\", \"remainingLife
\": 99.8, \"testMode\": false}"
        }
    ]
}
```

10. Each device type has a set of invokable device methods. To list the available methods for your device type, run the **get-device-methods** command, as follows.

```
aws iot1click-devices get-device-methods --device-id G030PM0123456789
{
    "DeviceMethods": [
        {
            "MethodName": "getDeviceHealthParameters"
        },
        {
            "MethodName": "setDeviceHealthMonitorCallback"
        },
        {
            "MethodName": "getDeviceHealthMonitorCallback"
        },
        {
            "MethodName": "setOnClickCallback"
        },
        {
            "MethodName": "getOnClickCallback"
        }
    ]
}
```

To invoke one of the available methods, use the **invoke-device-method** command, as shown next.

```
aws iot1click-devices invoke-device-method --cli-input-json file://invoke-device-
method.json
{
    "DeviceMethodResponse": "{\"remainingLife\": 99.8}"
}
```

Here, `invoke-device-method.json` contains the following.

```
{
    "DeviceId": "G030PM0123456789",
    "DeviceMethod": {
        "DeviceType": "device",
        "MethodName": "getDeviceHealthParameters"
    }
}
```

Note
The `get` methods (such as `getDeviceHealthParameters`) don't expect parameters. Therefore, the `"DeviceMethodParameters": ""` line within the JSON file can't be used (doing so will cause this: `An error occurred (InvalidRequestException) when calling the InvokeDeviceMethod operation: A request parameter was invalid.`)

11. By running **aws iot1click-devices list-devices --device-type button**, you can see that the default value for `Enabled` is `false`. The following command sets this key to `true`.

```
aws iot1click-devices update-device-state --device-id G030PM0123456789 --enabled
```

To set it back to `false`, run the previous command, again using the **--no-enabled** argument.

12. If customer information changes, you can update a device's placement information as shown next (notice the switch from **iot1click-devices** to **iot1click-projects**). Run the following command to view **customer217**'s current information (see `attributes`).

```
aws iot1click-projects describe-placement --project-name SeattleDumpsters --placement-
name customer217
```

```
{
    "placement": {
        "projectName": "SeattleDumpsters",
        "placementName": "customer217",
        "attributes": {
            "phone": "206-123-4567",
            "location": "1800 9th Ave Seattle, WA 98101"
        },
        "createdDate": 1563488454,
        "updatedDate": 1563488454
    }
}
```

Next, run the following command to update the customer's phone and location attributes.

```
aws iot1click-projects update-placement --cli-input-json file://update-placement.json
```

Here **update-placement.json** contains the following.

```
{
    "projectName": "SeattleDumpsters",
    "placementName": "customer217",
    "attributes": {
        "phone": "206-266-1000",
        "location": "410 Terry Ave N Seattle, WA 98109"
    }
}
```

To review this update, run **describe-placement** again, as shown.

```
aws iot1click-projects describe-placement --project-name SeattleDumpsters --placement-
name customer217
{
    "placement": {
        "projectName": "SeattleDumpsters",
        "placementName": "customer217",
        "attributes": {
            "phone": "206-266-1000",
            "location": "410 Terry Ave N Seattle, WA 98109"
        },
        "createdDate": 1563488454,
        "updatedDate": 1563572842
    }
}
```

13. To update project information, use the **update-project** command. A project generally contains multiple customer placements. Here is the existing `SeattleDumpster` project information.

```
aws iot1click-projects describe-project --project-name SeattleDumpsters
{
    "project": {
        "arn": "arn:aws:iot1click:us-west-2:012345678901:projects/SeattleDumpsters",
        "projectName": "SeattleDumpsters",
        "description": "All dumpsters in the Seattle region.",
        "createdDate": 1563483100,
        "updatedDate": 1563483100,
        "placementTemplate": {
            "defaultAttributes": {
                "City": "Seattle"
            },
            "deviceTemplates": {
```

```
                "empty-dumpster-request": {
                    "deviceType": "button",
                    "callbackOverrides": {}
                }
            }
        },
        "tags": {}
    }
}
```

To change "All dumpsters in the Seattle region" to "All dumpsters (yard waster, recycling, and garbage) in the Seattle region", run the following command.

```
aws iot1click-projects update-project --project-name SeattleDumpsters --description
 "All dumpsters (yard waste, recycling, garbage) in the Seattle region."
```

You can see that the value of the `"description"` key has been updated for all `SeattleDumpsters` placements.

```
aws iot1click-projects describe-project --project-name SeattleDumpsters
{
    "project": {
        "arn": "arn:aws:iot1click:us-west-2:012345678901:projects/SeattleDumpsters",
        "projectName": "SeattleDumpsters",
        "description": "All dumpsters (yard waste, recycling, garbage) in the Seattle
 region.",
        "createdDate": 1563483100,
        "updatedDate": 1563819039,
        "placementTemplate": {
            "defaultAttributes": {
                "City": "Seattle"
            },
            "deviceTemplates": {
                "empty-dumpster-request": {
                    "deviceType": "button",
                    "callbackOverrides": {}
                }
            }
        },
        "tags": {}
    }
}
```

14. You can use tags to apply meta-information to project resources (**iot1click-projects**) and placement resources (**iot1click-devices**), as follows.

```
aws iot1click-projects tag-resource --cli-input-json file://projects-tag-resource.json
```

Here `projects-tag-resource.json` contains the following.

```
{
    "resourceArn": "arn:aws:iot1click:us-west-2:012345678901:projects/
SeattleDumpsters",
    "tags": {
        "Account": "45215",
        "Manager": "Tom Jones"
    }
}
```

To list the tags for the project resource, run the following.

```
aws iot1click-projects list-tags-for-resource --resource-arn "arn:aws:iot1click:us-
west-2:012345678901:projects/SeattleDumpsters"
{
    "tags": {
        "Manager": "Tom Jones",
        "Account": "45215"
    }
}
```

To see the project tags in context, run the following.

```
aws iot1click-projects describe-project --project-name SeattleDumpsters
{
    "project": {
        "arn": "arn:aws:iot1click:us-west-2:012345678901:projects/SeattleDumpsters",
        "projectName": "SeattleDumpsters",
        "description": "All dumpsters (yard waste, recycling, garbage) in the Seattle
 region.",
        "createdDate": 1563483100,
        "updatedDate": 1563819039,
        "placementTemplate": {
            "defaultAttributes": {
                "City": "Seattle"
            },
            "deviceTemplates": {
                "empty-dumpster-request": {
                    "deviceType": "button",
                    "callbackOverrides": {}
                }
            }
        },
        "tags": {
            "Manager": "Tom Jones",
            "Account": "45215"
        }
    }
}
```

To discover device Amazon Resource Names (ARN)s, run the following.

```
aws iot1click-devices list-devices
{
    "Devices": [
        {
            "Arn": "arn:aws:iot1click:us-west-2:012345678901:devices/G030PM0123456789",
            "Attributes": {
                "projectRegion": "us-west-2",
                "projectName": "SeattleDumpsters",
                "placementName": "customer217",
                "deviceTemplateName": "empty-dumpster-request"
            },
            "DeviceId": "G030PM0123456789",
            "Enabled": true,
            "RemainingLife": 99.7,
            "Type": "button",
            "Tags": {}
        }
    ]
}
```

To add tags to the previous device, run the following.

```
aws iot1click-devices tag-resource --cli-input-json file://devices-tag-resource.json
```

Here **devices-tag-resources.json** contains the following (notice the required casing of `ResourceArn` and `Tags`).

```
{
    "ResourceArn": "arn:aws:iot1click:us-west-2:012345678901:devices/G030PM0123456789",
    "Tags": {
        "Driver": "John Smith",
        "Driver Phone": "206-123-4567"
    }
}
```

To list the tags for the device resource, run the following.

```
aws iot1click-devices list-tags-for-resource --resource-arn "arn:aws:iot1click:us-
west-2:012345678901:devices/G030PM0123456789"
{
    "Tags": {
        "Driver Phone": "206-123-4567",
        "Driver": "John Smith"
    }
}
```

To see the device tags in context, run **list-devices**.

```
aws iot1click-devices list-devices
{
    "Devices": [
        {
            "Arn": "arn:aws:iot1click:us-west-2:012345678901:devices/G030PM0123456789",
            "Attributes": {
                "projectRegion": "us-west-2",
                "projectName": "SeattleDumpsters",
                "placementName": "customer217",
                "deviceTemplateName": "empty-dumpster-request"
            },
            "DeviceId": "G030PM0123456789",
            "Enabled": true,
            "RemainingLife": 99.7,
            "Type": "button",
            "Tags": {
                "Driver Phone": "206-123-4567",
                "Driver": "John Smith"
            }
        }
    ]
}
```

15. At this point, you can associate an action with a press of the device's button, such as triggering an AWS Lambda function or sending an Amazon SNS message. You can do this easily using the AWS IoT 1-Click console (the AWS IoT 1-Click Programming Model (p. 10) is also an option). After the appropriate actions are associated with the device, you can take the device to the customer's location and connect it to the their Wi-Fi network by using the same procedure described in steps 1 and 2.

AWS IoT 1-Click device teardown

The following steps describe how to reverse (undo) the previous steps.

1. To untag a project resource, run the following command.

```
aws iot1click-projects untag-resource --resource-arn "arn:aws:iot1click:us-
west-2:012345678901:projects/SeattleDumpsters" --tag-keys "Manager"
```

   This removes the project's `Manager` tag, as shown next.

```
aws iot1click-projects describe-project --project-name SeattleDumpsters
{
    "project": {
        "arn": "arn:aws:iot1click:us-west-2:012345678901:projects/SeattleDumpsters",
        "projectName": "SeattleDumpsters",
        "description": "All dumpsters (yard waste, recycling, garbage) in the Seattle
 region.",
        "createdDate": 1563483100,
        "updatedDate": 1563819039,
        "placementTemplate": {
            "defaultAttributes": {
                "City": "Seattle"
            },
            "deviceTemplates": {
                "empty-dumpster-request": {
                    "deviceType": "button",
                    "callbackOverrides": {}
                }
            }
        },
        "tags": {
            "Account": "45215"
        }
    }
}
```

2. To untag a device resource, run the following command.

```
aws iot1click-devices untag-resource --resource-arn "arn:aws:iot1click:us-
west-2:012345678901:devices/G030PM0123456789" --tag-keys "Driver Phone" "Driver"
```

   This removes the device's tags, as shown next (notice the empty list `"Tags": {}`).

```
aws iot1click-devices list-devices
{
    "Devices": [
        {
            "Arn": "arn:aws:iot1click:us-west-2:012345678901:devices/G030PM0123456789",
            "Attributes": {
                "projectRegion": "us-west-2",
                "projectName": "SeattleDumpsters",
                "placementName": "customer217",
                "deviceTemplateName": "empty-dumpster-request"
            },
            "DeviceId": "G030PM0123456789",
            "Enabled": true,
            "RemainingLife": 99.7,
            "Type": "button",
            "Tags": {}
        }
```

```
    ]
}
```

3. To disassociate a device from a placement, run the following command.

```
aws iot1click-projects disassociate-device-from-placement --project-name
 SeattleDumpsters --placement-name customer217 --device-template-name empty-dumpster-
request
```

As you can see in the following, placement `customer217` no longer has a device associated with it.

```
aws iot1click-projects get-devices-in-placement --project-name SeattleDumpsters --
placement-name customer217
{
    "devices": {}
}
```

4. To delete a placement from a project, run the following command.

```
aws iot1click-projects delete-placement --project-name SeattleDumpsters --placement-
name customer217
```

As you can see in the following, project `SeattleDumpsters` has no placements because placement `customer217` was the only placement within `SeattleDumpsters`.

```
aws iot1click-projects list-placements --project-name SeattleDumpsters
{
    "placements": []
}
```

5. To delete a project, run the following command.

```
aws iot1click-projects delete-project --project-name SeattleDumpsters
```

As you can see in the following, all projects are removed because `SeattleDumpsters` was the only project associated with your AWS IoT 1-Click account.

```
aws iot1click-projects list-projects
{
    "projects": []
}
```

If, for example, you want to let a friend try out your device using their AWS account, you must first unclaim your device from your AWS IoT 1-Click account, as follows.

```
aws iot1click-devices unclaim-device --device-id G030PM0123456789
{
    "State": "UNCLAIMED"
}
```

The device can now be used with any AWS IoT 1-Click account.

AWS IoT 1-Click Appendix

This section provides additional AWS IoT 1-Click information as indicated by the following.

AWS IoT 1-Click Supported Devices

Product	Device type	Device ID prefix	To claim device	Purchase link	Device region[†]
Seeed IoT Button For AWS IoT (US, EU, & Japan)	Button	P5SJVQ (the first 6 digits of the device ID)	Enter the device ID in the AWS IoT 1-Click mobile app to configure Wi-Fi and to claim the device.	Seeed Studio Bazaar	US West (Oregon)
SORACOM LTE-M Button (Japan only)	Button	7MF6JK (the first 6 digits of the device ID)	Enter the device ID in the AWS IoT 1-Click mobile app to claim the device.	SORACOM	US West (Oregon)
AWS IoT Enterprise Button (US, EU, & Japan)	Button	G030PM (the first 6 digits of the device ID)	Enter the device ID in the AWS IoT 1-Click mobile app to configure Wi-Fi and to claim the device.	Discontinued	US West (Oregon)
AT&T LTE-M Button (US only)	Button	B9GHXT (the first 6 digits of the device ID)	Enter the claim code obtained when device(s) are purchased in the AWS IoT 1-Click mobile app or in the AWS IoT 1-Click console. Alternatively, you may enter the device ID in the AWS IoT 1-Click mobile app to claim the device.	Discontinued	US West (Oregon)

†For more information about device regions, see AWS IoT 1-Click Devices (p. 3).

Note
AWS IoT 1-Click does not support AWS IoT buttons whose device serial numbers (DSN) begin with G030JF, G030MD, and G030PT. To learn how to connect these buttons to the AWS IoT cloud (without using AWS IoT 1-Click), see Cloud Programmable Dash Button.

AWS IoT 1-Click Service Limits

- You can have a maximum of 5 device templates per placement template. This corresponds to 5 devices per placement.
- There is a maximum of 512 AWS IoT 1-Click projects per AWS Region per account.
- There is a maximum of 50 tags per AWS IoT 1-Click resource. Tags are key/value pairs (metadata) that can be used to manage a resource. For more information, see AWS Tagging Strategies.

Document History for Developer Guide

The following table describes the documentation for this release of AWS IoT 1-Click.

- **API version: latest**
- **Latest documentation update:** October 22, 2018

Change	Description	Date
Release	Initial release of the documentation.	May 14, 2018
Editorial	Editorial improvements.	May 31, 2018
Editorial	Updated supported device table.	October 22, 2018

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.