**Chapter 7**

**AWS Introduction (Video 7-3)**

Hi, I’m Alan Hawse, Senior Vice President of Technical Staff for Solutions and Software at Cypress Semiconductor. In this video, I'm going to introduce you to Amazon Web Services. After that, I'll show you how to interact with it from a WICED IoT device.

## Introduction

Amazon Web Services, or AWS, is a secure cloud services platform, offering compute power, database storage, content delivery and other functionality - which incidentally makes more money for Amazon than their retail operations.

AWS is built from a vast array of both virtual and actual servers and networks as well as a boatload of webserver software and administrative tools such as AWS IoT, Amazon Elastic Cloud which is a virtualized compute capability for rent, Amazon Lamda which allows you to send event driven tasks through the Cloud, Databases like Dynamo DB and Amazon Auroroa, the Amazon simple notification system which allows you to send SMS and email messages, and so on. More features are being added all the time to increase the power of what AWS can do for you.

In this chapter we are going to focus on AWS IoT which provides the basic Cloud services for IoT devices.

The AWS IoT Cloud service supports MQTT and HTTP access, **plus** a bunch of server-side functionality that provides:

* A virtual MQTT Message Broker.
* A virtual HTTP Server.
* A Thing Registry, which is a web interface to manage the access to your *things* – you know – the "T" in IoT.
* A web interface to manage the certificates and rules about *things* to help you handle security.
* A "shadow", which is an online cache of the most recent state of your *thing***.**
* A Rules Engine application that runs in the cloud that can subscribe to Topics and take programmatic actions based on messages – for example, you could configure it to subscribe to an "Alert" topic, and if a *thing* publishes a warning message to the alert topic, it uses Amazon SNS to send a SMS Text Message to your cell phone. Pretty cool, right?
* And an IoT applications SDK to help you build Web pages and cell phone Apps to go along with your IoT device.

## Resources

There are three types of resources in AWS: *Things*, *Certificates*, and *Policies*.

A *thing* is a representation of a 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 a physical entity that does not connect to AWS IoT but can be related to other devices that do (for example, a car that has engine sensors or a control panel).

Certificates are used by AWS to verify identities in both directions. That is, your IoT device validates that it is talking to AWS and AWS validates that your device is who it claims to be. Certificates for your devices can be created and managed on the AWS web site, which I'll demonstrate in a minute.

After creating a certificate for your internet-connected *thing*, you must create and attach an AWS IoT policy that will determine what AWS IoT operations the *thing* may perform. AWS IoT policies are JSON documents and they follow the same conventions as AWS Identity and Access Management policies. I'll show you more on this later.

## Console

AWS provides a web "console" at console.aws.amazon.com that allows you to create resources, interact with thing shadows, publish test messages, subscribe to updates, and do other IoTish things. When I teach this class in person, I provide access to my own AWS account but since you are watching this as a video, you will need to create your own account to try out the exercises. Don't worry – a basic account is free for a year.

Once you get logged into the console, select IoT Core. On the left side you see a menu with some activities that you can do. For example, if I select Manage, I can see a list of all the things that have been created on my account.

Clicking on a thing will show me its details like its resource name, its security certificate, its shadow state, and so on.

Going back, I can see all the certificates that I have created by clicking on secure. Then if I choose a certificate I can see its policy and which things it is attached to.

Back to the top again, I can look at settings to see my MQTT endpoint. This is my virtual MQTT broker that I will connect to from my IoT device.

NOTE: Is it safe to show the account name, MQTT broker endpoint, Thing resource names, etc? I think so because they won't have the certificates or passwords to access anything, but I want to make sure we are not opening ourselves up to attack.

The Test selection opens a virtual MQTT client that I can use to publish messages to a topic or subscribe to a topic. I'll show you how to use this once we have an MQTT device that we want to test out.

Finally, there is a selection that says "Learn". I would recommend that you go there now and watch the tutorial by clicking "Start the tutorial". It only takes a few minutes, but it gives you a good idea of how the pieces fit together.

## Shadows

A *thing* shadow (sometimes referred to as a device shadow) is a JSON document that is used to store and retrieve current state information for a *thing* (device, app, and so on). The *Thing* Shadows service maintains a *thing* shadow for each *thing* you connect to AWS IoT. You can use *thing* shadows to get and set the state of a *thing* over MQTT or HTTP, regardless of whether the *thing* is currently connected to the Internet. Each *thing* shadow is uniquely identified by its name.

A JSON Shadow document looks like this….

NOTE: The screen will show a Shadow document here.

As you can see, the top levels are state, metadata, version, clientToken, and timestamp.

Most of the time you will only care about the state. It has 2 sub levels called desired and reported. Applications will change the values under "desired" to request an update to the state of a thing without having to directly connect to it. On the other side, Things will change the values under "reported" to keep track of their actual state.

This is useful because it means two devices don't need to be simultaneously connected to the Cloud to be able to communicate. For example, consider an IoT connected light bulb. If the power goes out at your house, the bulb will lose connectivity. Once the power goes back on, the light bulb can connect again and read the desired state to remember if it is supposed to be on or off.

The entries under desired and reported can be anything you want. In this example, there is a state called color and another state called sequence that indicates a color sequence. The names of the states and what they mean are entirely up to your application.

## Topics

Topics are the way you Publish and Subscribe to MQTT messages. You can think of them as an address for a message.

The AWS Message Broker will allow you to create Topics with almost any name, with one exception: Topic names that start with "$aws/" are reserved by AWS IoT for specific functions.

The main use of $aws/ Topics is to interact with Thing Shadows. Topics that look like this will interact with the shadow for a thing called Thing1:

NOTE: These will show on screen. They are not intended to be read aloud.

$aws/things/Thing1/shadow/update

$aws/things/Thing1/shadow/update/accepted

$aws/things/Thing1/shadow/update/rejected

$aws/things/Thing1/shadow/update/documents

$aws/things/Thing1/shadow/get

$aws/things/Thing1/shadow/get/accepted

$aws/things/Thing1/shadow/get/rejected

$aws/things/Thing1/shadow/delete

The first one is the Topic you would Publish to in order to send a new value to the Shadow for Thing1. The message you send is just a JSON message with the value or values that you want to change. For example, you could publish this message to say that you want the color to be blue.

NOTE: This will just show up on screen.

{"state" : {"desired" : {"color" : "BLUE"}}}

After sending your message, the server will publish a message to either the update/accepted or update/rejected topic depending on if your message was accepted and put into the Shadow. If your message was accepted, the server will also publish the complete updated Shadow document to update/documents.

The next group are used when you want to get the state of a shadow. Publishing a message to get will cause the server to respond with either get/accepted or get/rejected. In the case of get/accepted the message it sends will be the complete Shadow document.

Finally, you can use /delete to remove a Shadow document completely. There are a few other Shadow Topics available and I cover them in the manual.

That's it for my introduction to AWS. In the next video, I'll show you how to create things, certificates and policies.

As always, you can post your comments and questions in our Wifi developer community or you are welcome to email me at alan\_hawse@cypress.com or tweet me @askioexpert with your comments, suggestions, criticisms and questions.