**Chapter 7**

**Cloud Introduction (Video 7-0)**

Hello, I’m Alan Hawse. Welcome to Chapter 7 of the Cypress Academy WICED WiFi 101 course. In the previous chapter we talked about how data is sent over WiFi using TCP/IP sockets. That's really good to understand, but when you really want to talk to the Cloud you always use a higher-level Cloud protocol such as HTTP, MQTT, AMQP or CoAP. In this chapter, that's what we're going to talk about.

Since this topic covers such a broad range, I will split it into 5 sub-chapters: Chapter 7A will cover the overall Cloud at a high level. In chapter 7B I'll talk in detail about HTTP and how it is used inside of WICED. In chapter 7C I'll cover MQTT – specifically, using MQTT to communicate with Amazon's Web Services Cloud – otherwise known as AWS. In the final chapters I will cover AMQP and CoAP.

So, what is "the Cloud"? Put simply, it's the name we use for that giant amalgamation of all the stuff you need to provide Web sites and all the other network-based services to get IoT working.

Why do you need the Cloud? Well, when you try to service a very large number of people and devices you have a very difficult and expensive problem. To have a fast and always available and always reliable system you need to have enough networks, disk drives, computers, people, databases, and all of that to make it all run. The solution to this problem is a standardized, shared, scalable system which we all know as the Cloud.

The term "the Cloud" generally includes:

1. Networks
2. Storage
3. Servers
4. Scalability
5. Load Balancing
6. Fault Tolerance
7. Management Tools
8. Software
9. Databases

Obviously, you can build your own cloud, and in fact many people do. But there's a reason why Jeff Bezos is the richest man in the world and that is because Amazon is really really good at cloud services.

So, how do you hook up to the cloud? Well, there's really just four main application layer protocols used to access the Cloud. Specifically: HTTP, MQTT, AMQP, and CoAP. WICED WiFi supports all four of them and I'll talk about each one of them briefly now and then we'll dive into the details in chapters 7B through the end.

**HTTP**

The original and most common Cloud protocol is HTTP, which stands for Hyper Text Transfer Protocol. It's a text-based protocol that operates over TCP sockets.

It can perform 9 functions – also called methods or verbs - as shown here, but most of the time an IoT device will use the "GET" method to request data from the cloud and the "POST" method to send data to the cloud.

1. GET
2. PUT
3. POST
4. HEAD
5. PATCH
6. DELETE
7. CONNECT
8. OPTIONS
9. TRACE

To use any of these commands, the way it works is you open up a TCP socket, typically to port 80 – or potentially to port 443 for secure HTTP which is denoted as HTTPS – then you send a request, and then the server sends a response. All HTTP communication is done in this way – a client – which can be an IoT device or, as you've used a lot I'm sure, a web browser - sends a request to the server and then the server responds with a text-based response.

The requests and responses are text-based strings that contain various types of data such as HTML, JSON, JPEG images, and so on, and I'll show you exactly what the requests and the responses look like in the next set of videos.

It's possible - and actually very common - to build IoT devices that use HTTP to "POST" and "PUT" data from the Cloud. However, HTTP has quite a bit of overhead, so it's slowly being displaced by other protocols that are more suited for use in the IoT. The reason that HTTP is so wide spread is it was the original Cloud protocol and there's a significant amount of infrastructure build up around it that supports it.

**MQTT**

The second protocol I'll talk about is MQTT, which stands for Message Queueing Telemetry Transport. It's a lightweight protocol that allows a device to Publish Messages to a specific Topic on a Message Broker. The Message Broker will then relay the messages to all of devices that are Subscribed to that Topic.

MQTT doesn't dictate the message format, but the de-facto standard for MQTT messages is JSON which – remember - we discussed back in chapter 4. Don't worry – I'm going to give you lots of practice using JSON coming up in a later video.

A Topic is just the name of a message queue on the Broker such as mydevice/status or mydevice/temperature. The name can be just about anything you want but by convention, hierarchical names are commonly used and they're denoted by slashes in the name.

Publishing is the process by which the client sends a message to a specific Topic on a message broker.

A Subscription is just a request by a device to have all of the messages Published to a specific Topic relayed to that client which is subscribed to the Topic.

The Message Broker is a server that responds to requests from clients to: establish and tear down connections, add and remove subscriptions, and accept messages. The Message Broker also handles forwarding messages to a topic to any clients that have subscribed to that topic.

MQTT provides 3 levels of Quality of Service, commonly known as QOS. They are:

* Level 0 which is at most once, meaning messages are delivered once or possibly not delivered at all.
* Level 1 is at least once, meaning the messages are certain to be delivered, but may be possibly sent multiple times.
* Level 2 means exactly once, meaning every message will be delivered exactly one time.

Where HTTP is a one to one relationship protocol - a client talks to a server. MQTT on the other hand is a many to many relationship which gives you all kinds of interesting options to build your IoT systems.

As with HTTP, MQTT supports non-secure as well as secure connections, but instead of 80 and 443 it provides them on 1883 and 8883 respectively.

There are a bunch of cloud providers that support MQTT including Amazon AWS as well as IBM BlueMix.

**AMQP**

Next, let's talk about AMQP. It stands for Advanced Message Queuing Protocol.

AMQP is a binary application layer protocol designed to support a wide variety of messaging applications and communication patterns. It provides flow controlled, message-oriented communication with message-delivery guarantees just like MQTT such as at-most-once, at-least-once and exactly-once. It also provides for authentication and encryption based on SSL or TLS. The Wikipedia article on AMQP is outstanding.

The AMQP specification is defined in several layers: (i) a type system, (ii) a symmetric, asynchronous protocol for the transfer of messages from one process to another, (iii) a standard, extensible message format and (iv) a set of standardized but extensible 'messaging capabilities.'

Cloud providers that support AMQP include Microsoft Windows Azure, VMWare, and RedHat.

**CoAP**

Finally, I'll talk about CoAP, which stands for Constrained Application Protocol. What is it with these people and all the acronyms? Seriously, are they driving you crazy? I certainly am. Anyway.

CoAP makes use of two message types, requests and responses, using a simple, binary, base header format. The base header may be followed by options in an optimized Type-Length-Value format. Unlike the previous protocols, CoAP is typically built on UDP and optionally DTLS (remember, DTLS is secure UDP).

Any bytes after the header in the packet are considered the message body, if there is going to be any. The length of the message body is implied by the length of the UDP datagram. When bound to UDP the entire message MUST fit within a single datagram.

It's also possible to use HTTP messages as a transport layer for CoAP.

Cloud providers that use CoAP include Samsung ARTIK.

**Wrap-Up**

OK, that's it for the introduction to the Cloud. In the next video we'll get down to the nitty-gritty details of HTTP, and I'll show you how to build WICED HTTP applications.

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 questions, comments, suggestions, and criticisms. Thank you!