**Chapter 7**

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

Hi, I’m Alan Hawse, Senior Vice President of Technical Staff for Solutions and Software at Cypress Semiconductor. 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 good to understand, but when you really talk to the Cloud you usually use a higher-level Cloud protocol such as HTTP, MQTT, AMQP or CoAP. In this chapter, that's what we'll talk about.

Since this topic covers such a broad range, we will split it into 3 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 in WICED. Finally, in chapter 7C we'll cover MQTT – specifically, using MQTT to communicate with Amazon's Web Services Cloud – otherwise known as AWS.

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

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

The term "the Clould" generally includes:

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

There are four main application layer protocols used to access the Cloud. They are: HTTP, MQTT, AMQP, and CoAP. I'll talk about each one very briefly now and then dive into details of HTTP and MQTT in chapters 7B and 7C.

**HTPP**

The first protocol is HTTP, which stands for Hyper Text Transfer Protocol. It is 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 "GET" to request data from the cloud and "POST" to send data to the cloud.

To use any of these commands, you open a TCP socket typically to port 80 – or port 443 for secure HTTP which is denoted as HTTPS – then you send a request, and the server sends a response. All HTTP communication is done this way – a client – which can be an IoT device, a web browser, etc. - sends a request and the server sends a response.

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

It is possible (and semi-common) to build IoT devices that use HTTP to "POST" and "PUT" data to/from the Cloud. However, HTTP has a lot of overhead, so it is slowly being displaced by other protocols that are more suited to IoT.

**MQTT**

The second protocol I'll talk about is MQTT, which stands for Message Queueing Telemetry Transport. It is 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 message to all devices that are Subscribed to that Topic.

MQTT doesn't dictate the message format, but the de-facto standard is JSON, which we discussed back in chapter 4. Don't worry – you'll get lots of practice using JSON coming up.

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 hierarchy is denoted by slashes in the name.

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

A Subscription is a request by a device to have all messages Published to a specific topic related to the client.

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 or QOS. They are:

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

MQTT operates on TCP Ports 1883 for non-secure and 8883 for secure (TLS).

Cloud providers that support MQTT include Amazon AWS and 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 efficiently support a wide variety of messaging applications and communication patterns. It provides flow controlled, message-oriented communication with message-delivery guarantees similar to MQTT such as *at-most-once*, *at-least-once* and *exactly-once*. It also provides authentication and encryption based on [SASL](https://en.wikipedia.org/wiki/Simple_Authentication_and_Security_Layer) or [TLS](https://en.wikipedia.org/wiki/Transport_Layer_Security). It assumes an underlying reliable transport layer protocol such as Transmission Control Protocol (TCP).

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 is Constrained Application Protocol.

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. CoAP is by default bound to UDP and optionally to [DTLS](https://en.wikipedia.org/wiki/DTLS), providing a high level of communications security.

Any bytes after the headers in the packet are considered the message body, if any. The length of the message body is implied by the datagram length. When bound to UDP the entire message MUST fit within a single datagram. When used with [6LoWPAN](https://en.wikipedia.org/wiki/6LoWPAN) as defined in [RFC 4944](https://tools.ietf.org/html/rfc4944), messages SHOULD fit into a single [IEEE 802.15.4](https://en.wikipedia.org/wiki/IEEE_802.15.4) frame to minimize fragmentation.

The mapping of CoAP with [HTTP](https://en.wikipedia.org/wiki/HTTP) is also defined, allowing proxies to be built providing access to CoAP resources via HTTP in a uniform way.

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 into the nitty-gritty details of HTTP.

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