# Chapter 7a: Application Layer Protocols for Cloud Connectivity

## Objective

At the end of Chapter 7 you will understand how to build a complete WICED IoT App using one of the cloud application protocols (MQTT, COAP, AMQP, HTTP or Sockets). In addition, you will have a big picture understanding of each of those protocols.

This section is the prequel to Chapter 7 with the foundation information required to understand the rest of the chapter. At the end of this section you should understand the basics of the Application Protocols HTTP, MQTT, AMQP, and COAP.

## Time: 4 Hours

## Fundamentals

### Application Layer Protocols

Recall in chapter 5 we talked about the TCP/IP networking stack. In the section on the stack we talked about using a layered model to simplify

### [Hyper Text Transfer Protocol (HTTP)](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)

HTTP is a text-based Application Layer Protocol that operates over TCP Sockets. It can perform the following functions:

* GET (retrieve data) from a specific place
* POST (put data) to a specific place
* As well as HEAD, PUT, DELETE, TRACE, OPTIONS, CONNECT, PATH (less commonly used)

To initiate these commands, you open a socket generally to TCP port 80 and send the text based command (CRLF terminated) and read the replies. This request/reply protocol is used for every command; replies are sent with a resulting Content-Type string which indicates the type of data encoding for the response. The content-type string uses a Multipurpose Internet Mail Extension (MIME) type to indicate the type of data being received (e.g. text/html or image/jpeg)

For instance, you can send an HTTP get request to open “/” on example.com:

GET /index.html HTTP/1.1

Host: [www.example.com](http://www.example.com)

Example.com will respond with:

HTTP/1.1 200 OK

Date: Mon, 23 May 2005 22:38:34 GMT

Content-Type: text/html; charset=UTF-8

Content-Encoding: UTF-8

Content-Length: 138

Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT

Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)

ETag: "3f80f-1b6-3e1cb03b"

Accept-Ranges: bytes

Connection: close

<html>

<head>

<title>An Example Page</title>

</head>

<body>

Hello World, this is a very simple HTML document.

</body>

</html>

It is possible (and semi-common) to build IoT devices that use HTTP to “PUT” their data to webservers in the cloud and “GET” their instructions/data from webservers. However, HTTP is somewhat heavy and is generally being displaced by other protocols that are more suited to IoT.

### [Message Queueing Telemetry Transport (MQTT)](https://en.wikipedia.org/wiki/MQTT)

MQTT is a lightweight messaging 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.

A Topic is simply the name of a message queue e.g. “mydevice/status” or “mydevice/pressure”.

A Subscription is just the request by a client to have all messages published to a specific topic sent to the client

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

A Message Broker is just a server that handles the tasks:

* Establishing connections (MQTT Connect)
* Tearing down connections (MQTT Disconnect)
* Accepting subscriptions to a Topic from clients (MQTT Subscribe)
* Turning off subscriptions (MQTT Unsubscribe)
* Accepting messages from clients and pushing them to the subscribers (MQTT Publish)

The format of the messages being sent in MQTT is unspecified. The message broker does not know (or care) anything about the format of the data and it is up to the system designer to specify an overall format of the data. All that being said, [JavaScript Object Notation (JSON)](https://en.wikipedia.org/wiki/JSON) has become the lingua franca of IoT.

Cloud providers that use MQTT include Amazon AWS,

### [Constrained Object Application Protocol (COAP)](https://en.wikipedia.org/wiki/Constrained_Application_Protocol)

Cloud providers that use COAP include Samsung ARTIK

### [Advanced Message Queuing Protocol (AMQP)](https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol)

Cloud providers that use AMQP include Microsoft (e.g. Windows Azure), VMWare, and Redhat.

### [JavaScript Object Notation (JSON)](https://en.wikipedia.org/wiki/JSON)

JSON is an open-standard format that uses human-readable text to transmit data consisting of attribute–value pairs. JSON supports the following data types

* Double precision floating point
* Strings
* Boolean (true or false)
* Array (use “[]” to specify the array)
* Key/Value pairs as “key”:value (use “{}” to specify the keymap)

For example, a legal JSON file looks like this:

{

“astringkey” : “alan”,

“age” : 48,

“badass” : true,

“children”, [“Anna”,”Nicholas],

“address” : {

“number”:”201”

“street”: “East Main Street”

“city”: “Lexington”,

“state”:”Kentucky”,

“zipcode”:40507

}

}

The WICED-SDK has a JSON parser built in. You can find these function in the directory “Utilities🡪JSON\_parser”

## Further Reading

[1] RFC1700 – “Assigned Numbers”; Internet Engineering Task Force (IETF) - https://www.ietf.org/rfc/rfc1700.txt

[2] RFC2045 – “Multipurpose Internet Mail Extensions”; Internet Engineering Task Force (IETF) - https://tools.ietf.org/html/rfc2045

[3] IANA Service Name and Port Registry - <http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>

[4] RFC2616 – “Hypertext Transfer Protocol (HTTP) “ ; Internet Engineering Task Force (IETF) - <https://tools.ietf.org/html/rfc2616>

[5] RFC7159 – “The Javascript Object Notation (JSON) Data Interchange Format”; Internet Engineering Task Force (IETF) - <https://tools.ietf.org/html/rfc7159>

[6] MQTT - <http://mqtt.org/>

[7] RFC7959 – “The Constrained Application Protocol (CoAP)” ; Internet Engineering Task Force (IETF) - <https://tools.ietf.org/html/rfc7252>

[8] AMQP - <http://www.amqp.org/>