Asynchronous vs synchronous client applications

The client library supports two modes of operation. These are referred to as **synchronous** and **asynchronous** modes. If your application calls **MQTTClient_setCallbacks()**, this puts the client into asynchronous mode, otherwise it operates in synchronous mode.

In synchronous mode, the client application runs on a single thread. Messages are published using the **MQTTClient_publish()** and **MQTTClient_publishMessage()** functions. To determine that a QoS1 or QoS2 (see **Quality of service**) message has been successfully delivered, the application must call the **MQTTClient_waitForCompletion()** function. An example showing synchronous publication is shown in **Synchronous publication example**. Receiving messages in synchronous mode uses the **MQTTClient_receive()** function. Client applications must call either **MQTTClient_receive()** or **MQTTClient_yield()** relatively frequently in order to allow processing of acknowledgements and the MQTT "pings" that keep the network connection to the server alive.

In asynchronous mode, the client application runs on several threads. The main program calls functions in the client library to publish and subscribe, just as for the synchronous mode**.**

Processing of handshaking and maintaining the network connection is performed in the background, however. Notifications of status and message reception are provided to the client application using callbacks registered with the library by the call

to **MQTTClient_setCallbacks()** (see **MQTTClient_messageArrived()**, **MQTTClient_connectionLost()** and **MQTTClient_deliveryComplete()**). This API is not thread safe however - it is not possible to call it from multiple threads without synchronization. You can use the MQTTAsync API for that