knolleary / pubsubclient Public

Code Issues 459 Pull requests 59 Actions Projects Wiki Security Ins

New issue Jump to bottom

esp8266 TLS example #833

Open rforro wants to merge 2 commits into `knolleary:master` from `rforro:master`

Conversation 0 Commits 2 Checks 0 Files changed 2

Changes from all commits File filter Conversations Jump to

187 examples/mqtt_esp8266_tls_auth/mqtt_esp8266_tls_auth.ino

```
@@ -0,0 +1,187 @@
+ /*
+  Basic ESP8266 MQTT over TLS example with authentication
+  This sketch demonstrates the capabilities of the pubsub library in combination
+  with the ESP8266 board/library.
+
+  Use this if you have CA certificate and you're using credentials for login.
+
+  It order to establish TLS connection with the mqtt server following steps are neccesary:
+   - MQTT servers CA certificate has to be defined (use your own!)
+   - defined CA certificate has to be set
+   - time has to be obtained from NTP, because of CA expiration date validation
+  It connects to an MQTT server then:
+   - publishes "hello world" to the topic "outTopic" every two seconds
+   - subscribes to the topic "inTopic", printing out any messages
+     it receives. NB - it assumes the received payloads are strings not binary
+   - If the first character of the topic "inTopic" is an 1, switch ON the ESP Led,
+     else switch it off
+  It will reconnect to the server if the connection is lost using a blocking
+  reconnect function. See the 'mqtt_reconnect_nonblocking' example for how to
+  achieve the same result without blocking the main loop.
+  To install the ESP8266 board, (using Arduino 1.6.4+):
+   - Add the following 3rd party board manager under "File -> Preferences -> Additional Boards Manager URLs":
+        http://arduino.esp8266.com/stable/package_esp8266com_index.json
+   - Open the "Tools -> Board -> Board Manager" and click install for the ESP8266"
+   - Select your ESP8266 in "Tools -> Board"
+ */
+
+ // The hardcoded certificate authority for this example.
+ // Don't use it on your own apps!!!!!
```

```
+ static const char ca_cert[] PROGMEM = R"EOF(
+ -----BEGIN CERTIFICATE-----
+ MIIC1TCCAb2gAwIBAgIJAMPt1Ms37+hLMA0GCSqGSIb3DQEBCwUAMCExCzAJBgNV
+ BAYTAlVTMRIwEAYDVQQDDAkxMjcuMC4wLjMwHhcNMTgwMzE0MDQyMTU0WhcNMjkw
+ NTMxMDQyMTU0WjAhMQswCQYDVQQGEwJVUzESMBAGA1UEAwwJMTI3LjAuMC4zMIIB
+ IjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxsa4qU/tlzN4YTcnn/I/ffsi
+ jOPc8QRcwClKzasIZNFEye4uThl+LGZWFIFb8X8Dc+xmmBaWlPJbqtphgFKStpar
+ DdduHSW1ud6Y1FVKxljo3UwCMrYm76Q/jNzXJvGs6Z1MDNsVZzGJaoqit2H2Hkvk
+ y+7kk3YbEDlcyVsLOw0zCKL4cd2DSNDyhIZxWo2a8Qn5IdjWAYtsTnW6MvLk/ya4
+ abNeRfSZwi+r37rqi9CIs++NpL5ynqkKKEMrbeLactWgHbWrZeaMyLpuUEL2GF+w
+ MRaAwaj7ERwT5gFJRqYwj6bbfIdx5PC7h7ucbyp272MbrDa6WNBCMwQO222t4wID
+ AQABoxAwDjAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQCmXfrC42nW
+ IpL3JDkB8YlB2QUvD9JdMp98xxo33+xE69Gov0e6984F1Gluao0p6sS7KF+q3YLS
+ 4hjnzuGzF9GJMimIB7NMQ20yXKfKpmKJ7YugMaKTDWDhHn5679mKVbLSQxHCUMEe
+ tEnMT93/UaDbWBjV6zu876q5vjPMYgDHODqO295ySaA71UkijaCn6UwKUT49286T
+ V9ZtzgabNGHXfklHgUPWoShyze+G3g29I1BR0qABoJI63zaNu8ua42v5g1RldxsW
+ X8yKI14mFOGxuvcygG8L2xxysW7Zq+9g+O7gW0Pm6RDYnUQmIwY83h1KFCtYCJdS
+ 2PgozwkkUNyP
+ -----END CERTIFICATE-----
+ )EOF";
+
+ #include <ESP8266WiFi.h>
+ #include <PubSubClient.h>
+
+ // Update these with values suitable for your network.
+
+ const char* ssid = ".......";
+ const char* password = ".......";
+ const char* mqtt_server = ".......";
+ const char* mqtt_user = ".......";
+ const char* mqtt_password = ".......";
+
+ BearSSL::WiFiClientSecure espClient;
+ PubSubClient client(espClient);
+ unsigned long lastMsg = 0;
+ #define MSG_BUFFER_SIZE (50)
+ char msg[MSG_BUFFER_SIZE];
+ int value = 0;
+
+ void setClock()
+ {
+   configTime(3 * 3600, 0, "pool.ntp.org", "time.nist.gov");
+
+   Serial.print("Waiting for NTP time sync: ");
+   time_t now = time(nullptr);
+   while (now < 8 * 3600 * 2) {
+     delay(500);
+     Serial.print(".");
+     now = time(nullptr);
+   }
+   Serial.println("");
+   struct tm timeinfo;
+   gmtime_r(&now, &timeinfo);
```

```
+   Serial.print("Current time: ");
+   Serial.print(asctime(&timeinfo));
+ }
+
+ void setup_wifi() {
+
+   delay(10);
+   // We start by connecting to a WiFi network
+   Serial.println();
+   Serial.print("Connecting to ");
+   Serial.println(ssid);
+
+   WiFi.mode(WIFI_STA);
+   WiFi.begin(ssid, password);
+
+   while (WiFi.status() != WL_CONNECTED) {
+     delay(500);
+     Serial.print(".");
+   }
+
+   randomSeed(micros());
+
+   Serial.println("");
+   Serial.println("WiFi connected");
+   Serial.println("IP address: ");
+   Serial.println(WiFi.localIP());
+ }
+
+ void callback(char* topic, byte* payload, unsigned int length) {
+   Serial.print("Message arrived [");
+   Serial.print(topic);
+   Serial.print("] ");
+   for (int i = 0; i < length; i++) {
+     Serial.print((char)payload[i]);
+   }
+   Serial.println();
+
+   // Switch on the LED if an 1 was received as first character
+   if ((char)payload[0] == '1') {
+     digitalWrite(BUILTIN_LED, LOW);   // Turn the LED on (Note that LOW is the voltage level
+     // but actually the LED is on; this is because
+     // it is active low on the ESP-01)
+   } else {
+     digitalWrite(BUILTIN_LED, HIGH);  // Turn the LED off by making the voltage HIGH
+   }
+
+ }
+
+ void reconnect() {
+   char err_buf[256];
+
+   // Loop until we're reconnected
+   while (!client.connected()) {
```

```
+     Serial.print("Attempting MQTT connection...");
+     // Create a random client ID
+     String clientId = "ESP8266Client-";
+     clientId += String(random(0xffff), HEX);
+     // Attempt to connect
+     if (client.connect(clientId.c_str(), mqtt_user, mqtt_password)) {
+       Serial.println("connected");
+       // Once connected, publish an announcement...
+       client.publish("outTopic", "hello world");
+       // ... and resubscribe
+       client.subscribe("inTopic");
+     } else {
+       Serial.print("failed, rc=");
+       Serial.println(client.state());
+       espClient.getLastSSLError(err_buf, sizeof(err_buf));
+       Serial.print("SSL error: ");
+       Serial.println(err_buf);
+       Serial.println(" try again in 5 seconds");
+       // Wait 5 seconds before retrying
+       delay(5000);
+     }
+   }
+ }
+
+ void setup() {
+   pinMode(BUILTIN_LED, OUTPUT);     // Initialize the BUILTIN_LED pin as an output
+   Serial.begin(115200);
+   BearSSL::X509List *serverTrustedCA = new BearSSL::X509List(ca_cert);
+   espClient.setTrustAnchors(serverTrustedCA);
+   setup_wifi();
+   setClock(); // Required for X.509 validation
+   client.setServer(mqtt_server, 8883);
+   client.setCallback(callback);
+ }
+
+ void loop() {
+
+   if (!client.connected()) {
+     reconnect();
+   }
+   client.loop();
+
+   unsigned long now = millis();
+   if (now - lastMsg > 2000) {
+     lastMsg = now;
+     ++value;
+     snprintf (msg, MSG_BUFFER_SIZE, "hello world #%ld", value);
+     Serial.print("Publish message: ");
+     Serial.println(msg);
+     client.publish("outTopic", msg);
+   }
+ }
```

242 examples/mqtt_esp8266_tls_certs/mqtt_esp8266_tls_certs.ino

```
@@ -0,0 +1,242 @@
+ /*
+  Basic ESP8266 MQTT over TLS example with client certificate authentication
+  This sketch demonstrates the capabilities of the pubsub library in combination
+  with the ESP8266 board/library.
+
+  Use this if you have CA certificate, client certificate and client key.
+
+  It order to establish TLS connection with the mqtt server following steps are neccesary:
+   - MQTT servers CA certificate has to be defined (use your own!)
+   - client certificate and client key obtained from MQTT server have to be defined (use your own!)
+   - both certificates and the key need to be set
+   - time has to be obtained from NTP, because of CA expiration date validation
+  It connects to an MQTT server then:
+   - publishes "hello world" to the topic "outTopic" every two seconds
+   - subscribes to the topic "inTopic", printing out any messages
+     it receives. NB - it assumes the received payloads are strings not binary
+   - If the first character of the topic "inTopic" is an 1, switch ON the ESP Led,
+     else switch it off
+  It will reconnect to the server if the connection is lost using a blocking
+  reconnect function. See the 'mqtt_reconnect_nonblocking' example for how to
+  achieve the same result without blocking the main loop.
+  To install the ESP8266 board, (using Arduino 1.6.4+):
+   - Add the following 3rd party board manager under "File -> Preferences -> Additional Boards Manager URLs":
+        http://arduino.esp8266.com/stable/package_esp8266com_index.json
+   - Open the "Tools -> Board -> Board Manager" and click install for the ESP8266"
+   - Select your ESP8266 in "Tools -> Board"
+ */
+
+ // The hardcoded certificate authority for this example.
+ // Don't use it on your own apps!!!!!
+ const char client_private_key[] PROGMEM = R"EOF(
+ -----BEGIN RSA PRIVATE KEY-----
+ MIIEowIBAAKCAQEAsRNVTvqP++YUh8NrbXwE83xVsDqcB3F76xcXNKFDERfVd2P/
+ LvyDovCcoQtT0UCRgPcxRp894EuPH/Ru6Z2Lu85sV//i7ce27tc2WRFSfuhlRxHP
+ LJWHxTl1CEfXp/owkECQ4MB3pw6Ekc16iTEPiezTG+T+mQ/BkiIwcIK6CMlpR9DI
+ eYUTqv0f9NrUfAjdBrqlEO2gpgFvLFrkDEU2ntAIc4aPOP7yDOym/xzfy6TiG8Wo
+ 7nlh6M97xTZGfbEPCH9rZDjo5istym1HzF5P+COq+OTSPscjFGXoi978o6hZwa7i
+ zxorg4h5a5lGnshRu2Gl+Ybfa14OwnIrv/yCswIDAQABAoIBAHxwgbsHCriTcEoY
+ Yx6F0VTrQ6ydA5mXfuYvS/eIfIE+pp1IgMScYEXZobjrJPQg1CA1l0NyFSHS97oV
+ JPy34sMQxcLx6KABgeVHCMJ/EeJtnv7a3SUP0GIhhsVS95Lsl8RIG4hWub+EzFVK
+ eZqAB9N9wr4Pp3wZPodbz37B38rb1QPyMFmQOLlHjKTOmoxsXhL2ot+R3+aLYSur
+ oPO1kQo7/d0UAZoy8h9OQN4a2EXvawh4O2EvFGbc5X/yXwAdEQ4NPp9VZhkNIRkV
+ +XZ3FcIqEVOploKtRF/tVBTz3g61/lFz21L9PMmV5y8tvSafr2SpJugGVmp2rrVQ
+ VNyGlIECgYEA10JSI5gmeCU3zK6kvOfBp54hY/5dDrSUpjKkMxpmm7WZQ6Il/k7A
+ hMcLeMzHiriT7WhRIXF8AOr2MoEkHkH3DhVNN4ccieVZx2SE5P5mVkItZGLrrpfU
+ dysR/ARAI1HYegGUiKacZtf9SrRavU0m7fOVOiYwbFRhjyX+MyuteYkCgYEA0pbz
+ 4ZosetScP68uZx1sGlTfkcqLl7i15DHk3gnj6jKlfhvC2MjeLMhNDtKeUAuY7rLQ
```

```
+ guZ0CCghWAv0Glh5eYdfIiPhgqFfX4P5F3Om4zQHVPYj8xHfHG4ZP7dKQTndrO1Q
+ fLdGDTQLVXabAUSp2YGrijC8J9idSW1pYClvF1sCgYEAjkDn41nzYkbGP1/Swnwu
+ AEWCL4Czoro32jVxScxSrugt5wJLNWp508VukWBTJhugtq3Pn9hNaJXeKbYqVkyl
+ pgrxwpZph7+nuxt0r5hnrO2C7eppcjIoWLB/7BorAKxf8REGReBFT7nBTBMwPBW2
+ el4U6h6+tXh2GJG1Eb/1nnECgYAydVb0THOx7rWNkNUGggc/++why61M6kYy6j2T
+ cj05BW+f2tkCBoctpcTI83BZb53yO8g4RS2yMqNirGKN2XspwmTqEjzbhv0KLt4F
+ X4GyWOoU0nFksXiLIFpOaQWSwWG7KJWrfGJ9kWXR0Xxsfl5QLoDCuNCsn3t4d43T
+ K7phlwKBgHDzF+50+/Wez3YHCy2a/HgSbHCpLQjkknvgwkOh1z7YitYBUm72HP8Z
+ Ge6b4wEfNuBdlZll/y9BQQOZJLFvJTE5t51X9klrkGrOb+Ftwr7eI/H5xgcadI52
+ tPYglR5fjuRF/wnt3oX9JlQ2RtSbs+3naXH8JoherHaqNn8UpH0t
+ -----END RSA PRIVATE KEY-----
+ )EOF";
+
+ const char client_cert[] PROGMEM = R"EOF(
+ -----BEGIN CERTIFICATE-----
+ MIIDTzCCAjcCCQDPXvMRYOpeuDANBgkqhkiG9w0BAQsFADCBpjESMBAGA1UEAwwJ
+ MTI3LjAuMC4xMQswCQYDVQQGEwJVUzElMCMGA1UECgwcTXkgT3duIENlcnRpZmlj
+ YXRlIEF1dGhvcml0eTEUMBIGA1UECAwLQXJkdWlub0xhbmQxFTATBgNVBAcMDEFy
+ ZHVpbm9WaWxsZTEVMBMGA1UECgwMRVNQODI2NlVzZXJzMRgwFgYDVQQLDA9FU1A4
+ MjY2LUFyZHVpbm8wHhcNMTgwMzE0MDQwMDAwWhcNMjkwMjI0MDQwMDAwWjAsMRYw
+ FAYDVQQKDA1NeSBTZXJ2ZXIgT3JnMRIwEAYDVQQDDAkxMjcuMC4wLjMwggEiMA0G
+ CSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCxE1VO+o/75hSHw2ttfATzfFWwOpwH
+ cXvrFxc0oUMRF9V3Y/8u/IOi8JyhC1PRQJGA9zFGnz3gS48f9G7pnYu7zmxX/+Lt
+ x7bu1zZZEVJ+6GVHEc8slYfFOXUIR9en+jCQQJDgwHenDoSRzXqJMQ+J7NMb5P6Z
+ D8GSIjBwgroIyWlH0Mh5hROq/R/02tR8CN0GuqUQ7aCmAW8sWuQMRTae0Ahzho84
+ /vIM7Kb/HN/LpOIbxajueWHoz3vFNkZ9sQ8If2tkOOjmKy3KbUfMXk/4I6r45NI+
+ xyMUZeiL3vyjqFnBruLPGiuDiHlrmUaeyFG7YaX5ht9rXg7Cciu//IKzAgMBAAEw
+ DQYJKoZIhvcNAQELBQADggEBAEnG+FNyNCOkBvzHiUpHHpScxZqM2f+XDcewJgeS
+ L6HkYEDIZZDNnd5gduSvkHpdJtWgsvJ7dJZL40w7Ba5sxpZHPIgKJGl9hzMkG+aA
+ z5GMkjys9h2xpQZx9KL3q7G6A+C0bll7ODZlwBtY07CFMykT4Mp2oMRrQKRucMSV
+ AB1mKujLAnMRKJ3NM89RQJH4GYiRps9y/HvM5lh7EIK/J0/nEZeJxY5hJngskPKb
+ oPPdmkR97kaQnll4KNsC3owVlHVU2fMftgYkgQLzyeWgzcNa39AF3B6JlcOzNyQY
+ seoK24dHmt6tWmn/sbxX7Aa6TL/4mVlFoOgcaTJyVaY/BrY=
+ -----END CERTIFICATE-----
+ )EOF";
+
+ static const char ca_cert[] PROGMEM = R"EOF(
+ -----BEGIN CERTIFICATE-----
+ MIIC1TCCAb2gAwIBAgIJAMPt1Ms37+hLMA0GCSqGSIb3DQEBCwUAMCExCzAJBgNV
+ BAYTAlVTMRIwEAYDVQQDDAkxMjcuMC4wLjMwHhcNMTgwMzE0MDQyMTU0WhcNMjkw
+ NTMxMDQyMTU0WjAhMQswCQYDVQQGEwJVUzESMBAGA1UEAwwJMTI3LjAuMC4zMIIB
+ IjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxsa4qU/tlzN4YTcnn/I/ffsi
+ jOPc8QRcwClKzasIZNFEye4uThl+LGZWFIFb8X8Dc+xmmBaWlPJbqtphgFKStpar
+ DdduHSW1ud6Y1FVKxljo3UwCMrYm76Q/jNzXJvGs6Z1MDNsVZzGJaoqit2H2Hkvk
+ y+7kk3YbEDlcyVsLOw0zCKL4cd2DSNDyhIZxWo2a8Qn5IdjWAYtsTnW6MvLk/ya4
+ abNeRfSZwi+r37rqi9CIs++NpL5ynqkKKEMrbeLactWgHbWrZeaMyLpuUEL2GF+w
+ MRaAwaj7ERwT5gFJRqYwj6bbfIdx5PC7h7ucbyp272MbrDa6WNBCMwQO222t4wID
+ AQABoxAwDjAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQCmXfrC42nW
+ IpL3JDkB8YlB2QUvD9JdMp98xxo33+xE69Gov0e6984F1Gluao0p6sS7KF+q3YLS
+ 4hjnzuGzF9GJMimIB7NMQ20yXKfKpmKJ7YugMaKTDWDhHn5679mKVbLSQxHCUMEe
+ tEnMT93/UaDbWBjV6zu876q5vjPMYgDHODqO295ySaA71UkijaCn6UwKUT49286T
+ V9ZtzgabNGHXfklHgUPWoShyze+G3g29I1BR0qABoJI63zaNu8ua42v5g1RldxsW
+ X8yKI14mFOGxuvcygG8L2xxysW7Zq+9g+O7gW0Pm6RDYnUQmIwY83h1KFCtYCJdS
```

```
+ 2PgozwkkUNyP
+ -----END CERTIFICATE-----
+ )EOF";
+
+ #include <ESP8266WiFi.h>
+ #include <PubSubClient.h>
+
+ // Update these with values suitable for your network.
+
+ const char* ssid = ".......";
+ const char* password = ".......";
+ const char* mqtt_server = ".......";
+
+ BearSSL::WiFiClientSecure espClient;
+ PubSubClient client(espClient);
+ unsigned long lastMsg = 0;
+ #define MSG_BUFFER_SIZE (50)
+ char msg[MSG_BUFFER_SIZE];
+ int value = 0;
+
+ void setClock()
+ {
+   configTime(3 * 3600, 0, "pool.ntp.org", "time.nist.gov");
+
+   Serial.print("Waiting for NTP time sync: ");
+   time_t now = time(nullptr);
+   while (now < 8 * 3600 * 2) {
+     delay(500);
+     Serial.print(".");
+     now = time(nullptr);
+   }
+   Serial.println("");
+   struct tm timeinfo;
+   gmtime_r(&now, &timeinfo);
+   Serial.print("Current time: ");
+   Serial.print(asctime(&timeinfo));
+ }
+
+ void setup_wifi() {
+
+   delay(10);
+   // We start by connecting to a WiFi network
+   Serial.println();
+   Serial.print("Connecting to ");
+   Serial.println(ssid);
+
+   WiFi.mode(WIFI_STA);
+   WiFi.begin(ssid, password);
+
+   while (WiFi.status() != WL_CONNECTED) {
+     delay(500);
+     Serial.print(".");
+   }
```

```
+
+   randomSeed(micros());
+
+   Serial.println("");
+   Serial.println("WiFi connected");
+   Serial.println("IP address: ");
+   Serial.println(WiFi.localIP());
+ }
+
+ void callback(char* topic, byte* payload, unsigned int length) {
+   Serial.print("Message arrived [");
+   Serial.print(topic);
+   Serial.print("] ");
+   for (int i = 0; i < length; i++) {
+     Serial.print((char)payload[i]);
+   }
+   Serial.println();
+
+   // Switch on the LED if an 1 was received as first character
+   if ((char)payload[0] == '1') {
+     digitalWrite(BUILTIN_LED, LOW);   // Turn the LED on (Note that LOW is the voltage level
+     // but actually the LED is on; this is because
+     // it is active low on the ESP-01)
+   } else {
+     digitalWrite(BUILTIN_LED, HIGH);  // Turn the LED off by making the voltage HIGH
+   }
+
+ }
+
+ void reconnect() {
+   char err_buf[256];
+
+   // Loop until we're reconnected
+   while (!client.connected()) {
+     Serial.print("Attempting MQTT connection...");
+     // Create a random client ID
+     String clientId = "ESP8266Client-";
+     clientId += String(random(0xffff), HEX);
+     // Attempt to connect
+     if (client.connect(clientId.c_str())) {
+       Serial.println("connected");
+       // Once connected, publish an announcement...
+       client.publish("outTopic", "hello world");
+       // ... and resubscribe
+       client.subscribe("inTopic");
+     } else {
+       Serial.print("failed, rc=");
+       Serial.println(client.state());
+       espClient.getLastSSLError(err_buf, sizeof(err_buf));
+       Serial.print("SSL error: ");
+       Serial.println(err_buf);
+       Serial.println(" try again in 5 seconds");
+       // Wait 5 seconds before retrying
```

```
+       delay(5000);
+     }
+   }
+ }
+
+ void setup() {
+   pinMode(BUILTIN_LED, OUTPUT);     // Initialize the BUILTIN_LED pin as an output
+   Serial.begin(115200);
+   BearSSL::X509List *serverTrustedCA = new BearSSL::X509List(ca_cert);
+   BearSSL::X509List *serverCertList = new BearSSL::X509List(client_cert);
+   BearSSL::PrivateKey *serverPrivKey = new BearSSL::PrivateKey(client_private_key);
+   espClient.setTrustAnchors(serverTrustedCA);
+   espClient.setClientRSACert(serverCertList, serverPrivKey);
+   setup_wifi();
+   setClock(); // Required for X.509 validation
+   client.setServer(mqtt_server, 8883);
+   client.setCallback(callback);
+ }
+
+ void loop() {
+
+   if (!client.connected()) {
+     reconnect();
+   }
+   client.loop();
+
+   unsigned long now = millis();
+   if (now - lastMsg > 2000) {
+     lastMsg = now;
+     ++value;
+     snprintf (msg, MSG_BUFFER_SIZE, "hello world #%ld", value);
+     Serial.print("Publish message: ");
+     Serial.println(msg);
+     client.publish("outTopic", msg);
+   }
+ }
```