# **Test Guide: Secure MQTT-Based Device Provisioning and OTA Update System**

**Date:** April 08, 2025

**Purpose**

This guide provides a step-by-step process to test the secure MQTT-based system for device provisioning and Over-The-Air (OTA) firmware updates using the **EMQX MQTT platform** and **MQTTX** client. It verifies the functionality of the Python server script and ESP32 firmware, ensuring secure communication, provisioning, and firmware updates.

## **Prerequisites**

* **Hardware:**
  + ESP32 board with SIM7600 modem module.
  + SIM card with active cellular data plan.
  + USB cable for programming and serial monitoring.
* **Software:**
  + Python 3.x installed on the server machine with required libraries (paho-mqtt, cryptography).
  + Arduino IDE or PlatformIO for uploading ESP32 firmware.
  + MQTTX installed (download from [mqttx.app](https://mqttx.app/)).
  + Serial monitor tool (e.g., Arduino Serial Monitor).
* **EMQX Setup:**
  + EMQX Cloud account with a deployed MQTT broker instance (e.g., u008dd8e.ala.dedicated.aws.emqxcloud.com).
  + TLS enabled (port 8883).
  + Username: ESP32, Password: 12345 (as per code defaults).
  + CA certificate (emqx\_ca.crt) downloaded from EMQX dashboard.
* **Files:**
  + Python server script (server.py).
  + ESP32 firmware source code.
  + OTA firmware binary file (e.g., firmware.bin) with magic byte 0xE9.

## **Test Environment Setup**

### **Step 1: Configure EMQX MQTT Broker**

1. Log in to your EMQX Cloud dashboard.
2. Verify your deployment details:
   * Host: u008dd8e.ala.dedicated.aws.emqxcloud.com
   * Port: 8883 (TLS)
   * Username: ESP32
   * Password: 12345
3. Download the CA certificate (emqx\_ca.crt) and place it in the Python script directory.
4. Ensure the broker is running and accessible.

### **Step 2: Set Up Python Server**

1. Open a terminal on your server machine.
2. Navigate to the directory containing server.py.
3. Install dependencies:  
   pip install paho-mqtt cryptography
4. Verify the script configuration:
   * BROKER, PORT, USERNAME, PASSWORD, and CA\_CERT match your EMQX settings.
5. Run the server:  
   python server.py
6. Observe the server public key output and connection confirmation:  
   Server Public Key: <PEM key>

Connected to MQTT broker

### **Step 3: Prepare ESP32 Device**

1. Connect the ESP32 to your computer via USB.
2. Open the Arduino IDE or PlatformIO.
3. Load the ESP32 firmware code.
4. Update the following in the code if needed:
   * mqtt\_server: u008dd8e.ala.dedicated.aws.emqxcloud.com
   * mqtt\_port: 8883
   * DEFAULT\_USERNAME: ESP32
   * DEFAULT\_PASSWORD: 12345
   * SERVER\_PUBLIC\_KEY\_PEM: Replace with the server’s public key from Step 2.
5. Upload the firmware to the ESP32.
6. Open the Serial Monitor (115200 baud) to observe boot logs.

### **Step 4: Install and Configure MQTTX**

1. Launch MQTTX on your computer.
2. Create a new connection:
   * Name: Test-EMQX
   * Host: u008dd8e.ala.dedicated.aws.emqxcloud.com
   * Port: 8883
   * Client ID: MQTTX\_Test\_<random>
   * Username: ESP32
   * Password: 12345
3. Enable SSL/TLS:
   * Go to SSL/TLS settings.
   * Upload the emqx\_ca.crt as the CA certificate.
4. Connect to the broker and verify the status turns green.

## **Test Scenarios**

### **Test 1: Device Provisioning**

**Objective:** Verify secure credential provisioning from server to ESP32.

1. **Start ESP32:**
   * Power on the ESP32 with the SIM7600 connected.
   * Monitor serial output for modem initialization and IMEI retrieval (e.g., Cleaned IMEI: 123456789012345).
2. **Observe Provisioning Request:**

In Serial Monitor, look for:  
 Sending: AT+CMQTTPUB=0,1,60

Published to dev\_pass\_req: UUID:<IMEI>:NONCE:<nonce>:PUBKEY:<pubkey>

* + In MQTTX, subscribe to dev\_pass\_req and confirm the message appears.

1. **Provide Credentials via Server:**
   * In the server terminal, when prompted:  
      Provide password for <IMEI> (min 12 chars):  
      Enter: 1234567890ab (12+ characters).
   * Verify server output:  
     Sent encrypted credentials for <IMEI>
2. **Check ESP32 Response:**
   * In Serial Monitor, confirm receipt and decryption:  
     Decrypted credentials: DEVICE\_ID:<IMEI>:USERNAME:ESP32\_<IMEI>:PASSWORD:1234567890ab
   * Ensure the ESP32 restarts MQTT with new credentials and enters STATE\_RUNNING.
3. **Validate with MQTTX:**
   * Subscribe to esp32\_status in MQTTX.
   * Confirm status messages from the ESP32 (e.g., OTA:PROGRESS or periodic updates).

### **Test 2: OTA Firmware Update**

**Objective:** Test secure OTA update delivery and application.

1. **Prepare Firmware:**
   * Ensure firmware.bin starts with magic byte 0xE9 (use a hex editor if needed).
   * Place it in the server directory.
2. **Initiate OTA from Server:**
   * In the server terminal, type cmd.
   * Enter:
     + Command: ota
     + IMEI: <ESP32 IMEI> (from Test 1)
     + File path: firmware.bin

Observe:  
 Starting OTA: File size: <size> bytes, Total chunks: <num>

Sent OTA:BEGIN

1. **Monitor ESP32:**

In Serial Monitor, verify:  
 Received message on OTA\_Update: OTA:BEGIN:<size>:<hash>:<signature>:DEVICE:<IMEI>

OTA:STARTED

* + Watch chunk processing:  
    Sent ack: OTA:PROGRESS:<chunk>:<received>/<total>:DEVICE:<IMEI>

1. **Track Progress in MQTTX:**
   * Subscribe to esp32\_status.
   * Confirm progress messages (e.g., OTA:PROGRESS:0:1024/50000:DEVICE:<IMEI>).
   * Verify final messages:  
     OTA:SUCCESS:PENDING\_VALIDATION
2. **Validate Update:**
   * After ESP32 restarts, check Serial Monitor:  
     Running from partition: ota\_1
   * Confirm new firmware behavior (e.g., log message changes if modified).

### **Test 3: Command Execution (Reset)**

**Objective:** Test command handling and confirmation.

1. **Send Reset Command from Server:**
   * In server terminal, type cmd.
   * Enter:
     + Command: reset soft
     + IMEI: <ESP32 IMEI>
   * Confirm:  
     Sent: RESET:SOFT:DEVICE:<IMEI>
2. **Verify ESP32 Response:**

In Serial Monitor:  
Received message on server\_cmd: RESET:SOFT:DEVICE:<IMEI>

Sent ack: RESET:SOFT:CONFIRMED:DEVICE:<IMEI>

Executing ESP.restart()

* + Observe reboot logs.

1. **Check MQTTX:**
   * Subscribe to esp32\_status.
   * Confirm RESET:SOFT:CONFIRMED:DEVICE:<IMEI> appears.

### **Test 4: Firmware Rollback**

**Objective:** Test reverting to previous firmware.

1. **Initiate Rollback:**
   * In server terminal, type cmd.
   * Enter:
     + Command: reverse
     + IMEI: <ESP32 IMEI>
   * Confirm:  
     Sent: reverse old firmware:DEVICE:<IMEI>
2. **Monitor ESP32:**

In Serial Monitor:  
 Received command to revert to previous firmware…

Successfully set boot partition to factory firmware: factory

OTA:REVERTED

* + After restart, verify:  
    Running from partition: factory

1. **Validate in MQTTX:**
   * Subscribe to esp32\_status.
   * Confirm OTA:REVERTED message.

## **Troubleshooting**

* **ESP32 Not Connecting:**
  + Check SIM card data plan and signal strength (AT+CSQ in Serial Monitor).
  + Verify MQTT credentials and CA certificate.
* **Server Not Responding:**
  + Ensure emqx\_ca.crt is in the correct directory.
  + Check firewall settings for port 8883.
* **OTA Fails:**
  + Confirm firmware.bin has magic byte 0xE9.
  + Check chunk acknowledgments in MQTTX; resend if missing.

## **Conclusion**

This guide tests the full lifecycle of the system: provisioning, OTA updates, command execution, and rollback. Successful completion confirms secure and reliable operation using EMQX and MQTTX. Log all results for reporting.