Autel Enterprise Device-to-Cloud Android MQTT Integration Guide

Overview

This MQTT client implements the following core functionalities:

- Automatic connection and reconnection mechanisms
- Device information management (gateway and drone)
- Automatic topic subscription
- Message publishing and receiving
- JSON message parsing
- Complete lifecycle management

---

1. Add Dependencies

Add the following dependencies to your app module's `build.gradle` file:

```
dependencies {
    // MQTT client library
    implementation 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.2.5'
    // Coroutine support
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.6.4'
}
```

---

2. Add Permissions

Add the following permissions to `AndroidManifest.xml`:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

---

3. Usage Instructions

3.1 Initialization and Connection

```
// In an Activity or Fragment
private val mqttHelper = MqttHelper()

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    // Initialize MQTT client
    mqttHelper.init()
    // Start connection with automatic retry mechanism
    mqttHelper.connectWithRetry()
}
```

3.2 Publish Messages

```
// Example of publishing a message
val message = JSONObject().apply {
    put("method", "get_info")
    put("data", JSONObject().put("device_id", "123"))
}.toString()

mqttHelper.publish("device/command", message)
```

3.3 Message Reception and Processing

The `MqttHelper` automatically handles the following types of messages:

- **Device Information Messages** (`device/info`)
- **Gateway-related Messages**
- **Drone-related Messages**

Example message format:

```
{
    "method": "gateway_info",
    "data": {
        "gateway_sn": "GW123456",
        "drone_sn": "DR789012"
    }
}
```

3.4 Topic Subscription

The system automatically subscribes to the following topics:

- **Device Information Topic**: `device/info`
- **Gateway Topics**:

```
thing/product/{gateway_sn}/osd
thing/product/{gateway_sn}/state
thing/product/{gateway_sn}/services_reply
thing/product/{gateway_sn}/events
thing/product/{gateway_sn}/requests
sys/product/{gateway_sn}/status
thing/product/{gateway_sn}/property/set_reply
thing/product/{gateway_sn}/get_token
```

- **Drone Topics**:

```
thing/product/{drone_sn}/osd
thing/product/{drone_sn}/state
```

3.5 Resource Cleanup

```
override fun onDestroy() {
    super.onDestroy()
    // Unsubscribe from all topics
    mqttHelper.unsubscribeAll()
    // Close the client
    mqttHelper.close()
}
```

---

4. Complete Example

MainActivity.kt

```
class MainActivity : AppCompatActivity() {
    private val mqttHelper = MqttHelper()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Initialize and connect
        mqttHelper.init()
        mqttHelper.connectWithRetry()

        // Send message example
        binding.sendButton.setOnClickListener {
            val message = JSONObject().apply {
                put("method", "get_info")
                put("data", JSONObject().put("device_id", "123"))
            }.toString()

            mqttHelper.publish("device/command", message)
        }
    }

    override fun onDestroy() {
        super.onDestroy()
```

```
        mqttHelper.unsubscribeAll()
        mqttHelper.close()
    }
}
```

---

5. Key Features

1. **Automatic Reconnection Mechanism**
    - Exponential backoff retry strategy
    - Maximum retry interval: 60 seconds
2. **Message Delivery Assurance**
    - Uses QoS 1 to ensure at-least-once delivery
    - Automatically handles connection loss scenarios
3. **Resource Management**
    - Automatically manages coroutine scopes
    - Completes subscription cancellation processes
    - Graceful shutdown handling
4. **Error Handling**
    - Comprehensive exception capture and logging
    - Connection state monitoring
    - Automatic reconnection mechanism

---

6. Best Practices

1. **Network Permissions**
    - Ensure the application has the necessary network permissions.
    - Check network connectivity status.
2. **Memory Management**
    - Call the `close()` method when components are destroyed.
    - Ensure all subscriptions are canceled.
3. **Message Handling**
    - Validate message format.
    - Handle JSON parsing exceptions properly.
4. **Debugging**
    - Use logcat for detailed debugging logs.
    - Look for logs with the tag "MQTT" to review critical events.

---

7. References

- Eclipse Paho Android Service
- MQTT Version 3.1.1
- Kotlin Coroutines