



APUSIC
固若长城
睿比世界

開發手冊

金蝶Apusic分佈式消息隊列V2.0.6_for_mqtt

版權所有 © 深圳市金蝶天燕雲計算股份有限公司2026。保留所有權利。

版權聲明

本文件所涉及的軟件著作權、版權等知識產權已依法進行了註冊，由金蝶天燕雲計算股份有限公司合法擁有。受《中華人民共和國著作權法》《計算機軟件保護條例》《知識產權保護條例》和相關國際版權條約、法律、法規以及其它知識產權法律和條約的保護。未經授權許可，不得非法使用。

免責聲明

本文件包含的版權資訊由金蝶天燕雲計算股份有限公司合法擁有，受法律的保護，金蝶天燕雲計算股份有限公司對本文件可能涉及到的非金蝶天燕雲計算股份有限公司的資訊不承擔任何責任。在法律允許的範圍內，您可以查閱並僅能夠在《中華人民共和國著作權法》規定的合法範圍內複製和打印本文件。任何單位和個人未經金蝶天燕雲計算股份有限公司書面授權許可，不得使用、修改、再發佈本文件的任何部分和內容，否則將被視為侵權，金蝶天燕雲計算股份有限公司有依法追究其責任的權利。

本文件如有更新，不另行通知。對本文件中的問題您可向金蝶天燕雲計算股份有限公司告知或查詢。未經本公司明確授予的任何權利均予保留。

商標聲明

 是深圳市金蝶天燕雲計算股份有限公司向中華人民共和國國家商標局申請註冊的註冊商標，註冊商標專用權由金蝶天燕合法擁有，受法律保護。未經金蝶天燕的書面許可，任何單位及個人不得以任何方式或理由對該商標的任何部分進行使用、複製、修改、傳播、抄錄或與其它產品捆綁使用銷售。凡侵犯金蝶天燕商標權的，金蝶天燕將依法追究其法律責任。本文件提及的其他所有商標或註冊商標，由各自的所有人擁有。

目錄

- 1 前言
 - 1.1 適用對象
 - 1.2 相關文檔
 - 1.3 技術支持
- 2 開發準備
 - 2.1 連接參數
 - 2.2 客戶端依賴
 - 2.2.1 Java
 - 2.2.2 Python
 - 2.2.3 JavaScript/Node.js
 - 2.2.4 Go
- 3 Java 開發示例
 - 3.1 生產者
 - 3.2 消費者
 - 3.3 遺囑消息
 - 3.4 會話保持
- 4 Python 開發示例
 - 4.1 生產者
 - 4.2 消費者
- 5 JavaScript/Node.js 開發示例
 - 5.1 生產者
 - 5.2 消費者
- 6 Go 開發示例
 - 6.1 生產者
 - 6.2 消費者
- 7 高級特性
 - 7.1 保留消息
 - 7.2 共享訂閱
 - 7.3 WebSocket 接入
 - 7.4 SSL/TLS 接入
- 8 最佳實踐
 - 8.1 ClientID 管理

- 8.2 QoS 選擇
- 8.3 會話管理
- 8.4 異常處理
- 8.5 性能優化

1 前言

本文檔為金蝶Apusic分佈式消息隊列for MQTT（Apusic Distributed Message Queue，簡稱：ADMQ for MQTT）的開發使用說明，幫助用戶快速學習如何使用金蝶Apusic分佈式消息隊列for MQTT進行開發。

1.1 適用對象

本文檔適用於IT信息化業務負責人、研發經理、軟件項目經理、軟件架構師、運維工程師。

1.2 相關文檔

瞭解更多ADMQ for MQTT產品相關的信息，請參閱以下ADMQ for MQTT產品手冊文檔集：

序號	手冊文檔	說明
1	金蝶Apusic分佈式消息隊列for MQTT快速使用手冊	簡單介紹瞭如何快速上手使用ADMQ for MQTT。
2	金蝶Apusic分佈式消息隊列for MQTT安裝手冊	詳細介紹如何在各操作系統上安裝ADMQ for MQTT，以及ADMQ for MQTT服務啟停等操作。
3	金蝶Apusic分佈式消息隊列for MQTT消息引擎用戶手冊	詳細介紹 ADMQ for MQTT 消息引擎相關功能的使用、配置、管理及配套工具的使用方法。
4	金蝶Apusic分佈式消息隊列for MQTT管控臺用戶手冊	詳細介紹ADMQ for MQTT管控臺相關功能的使用和操作說明。
5	金蝶Apusic分佈式消息隊列for MQTT開發手冊	詳細介紹基於各開發語言進行ADMQ for MQTT客戶端應用開發的說明。
6	金蝶Apusic分佈式消息隊列for MQTT遷移手冊	詳細介紹從MQTT Broker遷移到ADMQ for MQTT的說明。
7	金蝶Apusic分佈式消息隊列for MQTT運維手冊	詳細介紹ADMQ for MQTT的監控、運維、安全加固等運維說明。
8	金蝶Apusic分佈式消息隊列for MQTT性能優化手冊	詳細介紹ADMQ for MQTT性能調優的說明。

1.3 技術支持

ADMQ for MQTT產品提供全面的技術支持服務，您可以通過以下方式獲得技術支持：

- 網址: www.apusic.com
- 電話: 400-855-5800
- 郵箱: support@apusic.com
- 金蝶雲社區: <https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN>

您在取得技術支持時，請提供如下信息：

1. 您的姓名
2. 公司信息與聯繫方式
3. 操作系統及其版本
4. 產品版本號
5. 出現異常及錯誤的日誌、截圖等詳細信息

2 開發準備

2.1 連接參數

參數	說明	示例
Broker 地址	MQTT Broker 地址	localhost
端口	MQTT 端口	1883 (TCP) 、 8883 (SSL) 、 8083 (WebSocket)
ClientID	客戶端唯一標識	device-001
username	用戶名	device-service
password	密碼	your-password

2.2 客戶端依賴

2.2.1 Java

```
<dependency>
  <groupId>org.eclipse.paho</groupId>
  <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
  <version>1.2.5</version>
</dependency>
```

2.2.2 Python

```
pip install paho-mqtt
```

2.2.3 JavaScript/Node.js

```
npm install mqtt
```

2.2.4 Go

```
go get github.com/eclipse/paho.mqtt.golang
```

3 Java 開發示例

3.1 生產者

```
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;

public class MqttProducer {
    public static void main(String[] args) throws Exception {
        MqttClient client = new MqttClient("tcp://localhost:1883",
"publisher-001");

        MqttConnectOptions options = new MqttConnectOptions();
        options.setUsername("device-service");
        options.setPassword("your-password".toCharArray());
        options.setCleanSession(true);

        client.connect(options);

        String payload = "
{\"deviceId\":\"D001\", \"temperature\":24.5}";
        MqttMessage message = new MqttMessage(payload.getBytes("UTF-
8"));
        message.setQos(1);
        message.setRetained(false);

        client.publish("sensor/D001/temperature", message);
        System.out.println("Message sent");

        client.disconnect();
        client.close();
    }
}
```

3.2 消費者

```
import org.eclipse.paho.client.mqttv3.*;

public class MqttConsumer {
    public static void main(String[] args) throws Exception {
        MqttClient client = new MqttClient("tcp://localhost:1883",
"subscriber-001");

        MqttConnectOptions options = new MqttConnectOptions();
        options.setUsername("device-service");
        options.setPassword("your-password".toCharArray());
        options.setCleanSession(true);

        client.setCallback(new MqttCallback() {
            @Override
            public void connectionLost(Throwable cause) {
                System.out.println("Connection lost: " +
cause.getMessage());
            }

            @Override
            public void messageArrived(String topic, MqttMessage
message) throws Exception {
                System.out.println("Received: " + topic + " -> " +
new String(message.getPayload(), "UTF-8"));
            }

            @Override
            public void deliveryComplete(IMqttDeliveryToken token) {
                // 用于 QoS 1/2 的发布确认
            }
        });

        client.connect(options);
    }
}
```

```

        client.subscribe("sensor/+/temperature", 1);

        System.out.println("Subscribed");
    }
}

```

3.3 遺囑消息

```

MqttConnectOptions options = new MqttConnectOptions();
options.setUserName("device-service");
options.setPassword("your-password".toCharArray());

// 设置遺囑消息
options.setWill("device/D001/status", "offline".getBytes(), 1,
true);

client.connect(options);

// 上线后发送在线状态
client.publish("device/D001/status", "online".getBytes(), 1, true);

```

3.4 會話保持

```

MqttConnectOptions options = new MqttConnectOptions();
options.setUserName("device-service");
options.setPassword("your-password".toCharArray());
options.setCleanSession(false); // 保持會話

client.connect(options);
client.subscribe("sensor/D001/temperature", 1);

```

4 Python 開發示例

4.1 生產者

```
import paho.mqtt.client as mqtt

client = mqtt.Client(client_id="publisher-001")
client.username_pw_set("device-service", "your-password")

client.connect("localhost", 1883, 60)

payload = '{"deviceId":"D001","temperature":24.5}'
client.publish("sensor/D001/temperature", payload, qos=1,
retain=False)
print("Message sent")

client.disconnect()
```

4.2 消費者

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")
    client.subscribe("sensor+/temperature", qos=1)

def on_message(client, userdata, msg):
    print(f"Received: {msg.topic} -> {msg.payload.decode('utf-8')}")

client = mqtt.Client(client_id="subscriber-001")
client.username_pw_set("device-service", "your-password")
client.on_connect = on_connect
client.on_message = on_message
```

```
client.connect("localhost", 1883, 60)
client.loop_forever()
```

5 JavaScript/Node.js 開發示例

5.1 生產者

```
const mqtt = require('mqtt');

const client = mqtt.connect('mqtt://localhost:1883', {
  clientId: 'publisher-001',
  username: 'device-service',
  password: 'your-password'
});

client.on('connect', () => {
  const payload = JSON.stringify({ deviceId: 'D001', temperature:
24.5 });
  client.publish('sensor/D001/temperature', payload, { qos: 1 },
(err) => {
    if (err) {
      console.error('Publish error:', err);
    } else {
      console.log('Message sent');
    }
    client.end();
  });
});
```

5.2 消費者

```
const mqtt = require('mqtt');

const client = mqtt.connect('mqtt://localhost:1883', {
  clientId: 'subscriber-001',
  username: 'device-service',
```

```
    password: 'your-password'
  });

client.on('connect', () => {
  client.subscribe('sensor+/temperature', { qos: 1 }, (err) => {
    if (err) {
      console.error('Subscribe error:', err);
    } else {
      console.log('Subscribed');
    }
  });
});

client.on('message', (topic, message) => {
  console.log(`Received: ${topic} -> ${message.toString()}`);
});
```

6 Go 開發示例

6.1 生產者

```
package main

import (
    "fmt"
    "time"
    mqtt "github.com/eclipse/paho.mqtt.golang"
)

func main() {
    opts := mqtt.NewClientOptions().
        AddBroker("tcp://localhost:1883").
        SetClientID("publisher-001").
        SetUsername("device-service").
        SetPassword("your-password")

    client := mqtt.NewClient(opts)
    if token := client.Connect(); token.Wait() && token.Error() !=
nil {
        panic(token.Error())
    }

    payload := `{"deviceId":"D001","temperature":24.5}`
    token := client.Publish("sensor/D001/temperature", 1, false,
payload)
    token.Wait()

    fmt.Println("Message sent")
    client.Disconnect(250)
}
```

6.2 消費者

```
package main

import (
    "fmt"
    mqtt "github.com/eclipse/paho.mqtt.golang"
)

func main() {
    opts := mqtt.NewClientOptions().
        AddBroker("tcp://localhost:1883").
        SetClientID("subscriber-001").
        SetUsername("device-service").
        SetPassword("your-password")

    client := mqtt.NewClient(opts)
    if token := client.Connect(); token.Wait() && token.Error() !=
nil {
        panic(token.Error())
    }

    if token := client.Subscribe("sensor/+/temperature", 1,
func(client mqtt.Client, msg mqtt.Message) {
        fmt.Printf("Received: %s -> %s\n", msg.Topic(),
string(msg.Payload()))
    }); token.Wait() && token.Error() != nil {
        panic(token.Error())
    }

    select {}
}
```

7 高級特性

7.1 保留消息

```
MqttMessage message = new MqttMessage(payload.getBytes());
message.setQos(1);
message.setRetained(true); // 设置为保留消息
client.publish("sensor/D001/status", message);

// 清除保留消息
MqttMessage clearMessage = new MqttMessage(new byte[0]);
clearMessage.setRetained(true);
client.publish("sensor/D001/status", clearMessage);
```

7.2 共享訂閱

MQTT 5.0 共享訂閱格式：

```
$share/group1/sensor+/temperature
```

同一共享組內的訂閱者輪流接收消息，實現負載均衡。

7.3 WebSocket 接入

```
const client = mqtt.connect('ws://localhost:8083/mqtt', {
  clientId: 'web-client-001',
  username: 'device-service',
  password: 'your-password'
});
```

7.4 SSL/TLS 接入

```
MqttClient client = new MqttClient("ssl://localhost:8883",
    "publisher-001");

MqttConnectOptions options = new MqttConnectOptions();
options.setUserName("device-service");
options.setPassword("your-password".toCharArray());

// 配置 SSL 上下文
// options.setSocketFactory(sslSocketFactory);

client.connect(options);
```

8 最佳實踐

8.1 ClientID 管理

1. **全局唯一**：每個客戶端的 ClientID 必須在集群內唯一
2. **可讀性**：建議使用有意義的前綴，如 `device-`、`app-`、`service-`
3. **避免特殊字符**：只使用字母、數字、下劃線、連字符

8.2 QoS 選擇

場景	推薦 QoS
高頻 Telemetry 數據，可容忍丟失	QoS 0
一般業務消息	QoS 1
關鍵控制命令，要求恰好一次	QoS 2

8.3 會話管理

1. Clean Session = true：適用於不需要離線消息的場景，資源佔用少
2. Clean Session = false：適用於需要保持訂閱和接收離線消息的場景
3. MQTT 5.0：使用 Clean Start 和 Session Expiry Interval 精確控制會話行為

8.4 異常處理

1. **斷線重連**：實現連接丟失後的自動重連機制
2. **消息去重**：QoS 1 消息可能重複，業務側需實現幂等處理
3. **超時處理**：合理設置連接超時和 Keepalive 間隔

8.5 性能優化

1. **批量發送**：減少頻繁的小消息發送
2. **合理設置 Keepalive**：避免過短的心跳間隔增加網絡負載
3. **使用共享訂閱**：多個消費者均衡處理消息
4. **限制訂閱數量**：單個客戶端訂閱過多主題會影響性能
5. **避免通配符濫用**：儘量使用精確主題匹配

全國統一服務熱線
4008-555-800

金蝶天燕雲計算股份有限公司（簡稱“金蝶天燕”）成立於2000年，前身為“金蝶中間件公司”，是金蝶集團旗下基礎軟件平台提供商，雲計算國家標準制定企業，國家信創產業核心軟件企業。金蝶天燕是國家863重點研發計劃與核高基重大專項承接企業，也是“兩網一站四庫十二金”國家重點工程的基礎平台提供商，產品廣泛應用於政府、軍工、金融、能源等關鍵行業，累計服務客戶總數超過10萬家。



雲計算國家標準制定企業
金蝶集團旗下基礎軟件企業
資訊技術應用創新核心企業
官網：www.apusic.com

