Apusic 金蝶天燕

生成时间：2026-07-13


# 开发手册

金蝶Apusic分布式消息队列V2.0.6_for_mqtt

## 版权声明

本文档所涉及的软件著作权、版权等知识产权已依法进行了注册，由金蝶天燕云计算股份有限公司合法拥有。受《中华人民共和国著作权法》《计算机软件保护条例》《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

## 免责声明

本文档包含的版权信息由金蝶天燕云计算股份有限公司合法拥有，受法律的保护，金蝶天燕云计算股份有限公司对本文档可能涉及到的非金蝶天燕云计算股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经金蝶天燕云计算股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将被视为侵权，金蝶天燕云计算股份有限公司有依法追究其责任的权利。

本文档如有更新，不另行通知。对本文档中的问题您可向金蝶天燕云计算股份有限公司告知或查询。未经本公司明确授予的任何权利均予保留。

## 商标声明

Apusic是深圳市金蝶天燕云计算股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由金蝶天燕合法拥有，受法律保护。未经金蝶天燕的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯金蝶天燕商标权的，金蝶天燕将依法追究其法律责任。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

# 目录

# 1 前言

本文档为金蝶Apusic分布式消息队列for MQTT（Apusic Distributed Message Queue，简称：ADMQ for MQTT）的开发使用说明，帮助用户快速学习如何使用金蝶Apusic分布式消息队列for MQTT进行开发。

## 1.1 适用对象

本文档适用于IT信息化业务负责人、研发经理、软件项目经理、软件架构师、运维工程师。

## 1.2 相关文档

了解更多ADMQ for MQTT产品相关的信息，请参阅以下ADMQ for MQTT产品手册文档集：

| 序号 | 手册文档 | 说明 |
| --- | --- | --- |
| 1 | 金蝶Apusic分布式消息队列for MQTT快速使用手册 | 简单介绍了如何快速上手使用ADMQ for MQTT 。 |
| 2 | 金蝶Apusic分布式消息队列for MQTT安装手册 | 详细介绍如何在各操作系统上安装ADMQ for MQTT，以及ADMQ for MQTT服务启停等操作。 |
| 3 | 金蝶Apusic分布式消息队列for MQTT消息引擎用户手册 | 详细介绍 ADMQ for MQTT 消息引擎相关功能的使用、配置、管理及配套工具的使用方法。 |
| 4 | 金蝶Apusic分布式消息队列for MQTT管控台用户手册 | 详细介绍ADMQ for MQTT管控台相关功能的使用和操作说明。 |
| 5 | 金蝶Apusic分布式消息队列for MQTT开发手册 | 详细介绍基于各开发语言进行ADMQ for MQTT客户端应用开发的说明。 |
| 6 | 金蝶Apusic分布式消息队列for MQTT迁移手册 | 详细介绍从MQTT Broker迁移到ADMQ for MQTT的说明。 |
| 7 | 金蝶Apusic分布式消息队列for MQTT运维手册 | 详细介绍ADMQ for MQTT的监控、运维、安全加固等运维说明。 |
| 8 | 金蝶Apusic分布式消息队列for MQTT性能优化手册 | 详细介绍ADMQ for MQTT性能调优的说明。 |

## 1.3 技术支持

ADMQ for MQTT产品提供全面的技术支持服务，您可以通过以下方式获得技术支持：

- 网址：www.apusic.com
- 电话：400-855-5800
- 邮箱：support@apusic.com
- 金蝶云社区：https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN

您在取得技术支持时，请提供如下信息：

1. 您的姓名
2. 公司信息与联系方式
3. 操作系统及其版本
4. 产品版本号
5. 出现异常及错误的日志、截图等详细信息

# 2 开发准备

## 2.1 连接参数

| 参数 | 说明 | 示例 |
| --- | --- | --- |
| Broker 地址 | MQTT Broker 地址 | localhost |
| 端口 | MQTT 端口 | 1883（TCP）、8883（SSL）、8083（WebSocket） |
| ClientID | 客户端唯一标识 | device-001 |
| username | 用户名 | device-service |
| password | 密码 | your-password |

## 2.2 客户端依赖

### 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 开发示例

## 3.1 生产者

```
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 消费者

```
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 遗嘱消息

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

// 设置遗嘱消息
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 会话保持

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

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

# 4 Python 开发示例

## 4.1 生产者

```
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 消费者

```
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 开发示例

## 5.1 生产者

```
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 消费者

```
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 开发示例

## 6.1 生产者

```
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 消费者

```
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 高级特性

## 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 共享订阅

MQTT 5.0 共享订阅格式：

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

同一共享组内的订阅者轮流接收消息，实现负载均衡。

## 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 最佳实践

## 8.1 ClientID 管理

1. **全局唯一**：每个客户端的 ClientID 必须在集群内唯一
2. **可读性**：建议使用有意义的前缀，如 `device-`、`app-`、`service-`
3. **避免特殊字符**：只使用字母、数字、下划线、连字符

## 8.2 QoS 选择

| 场景 | 推荐 QoS |
| --- | --- |
| 高频 Telemetry 数据，可容忍丢失 | QoS 0 |
| 一般业务消息 | QoS 1 |
| 关键控制命令，要求恰好一次 | QoS 2 |

## 8.3 会话管理

1. Clean Session = true：适用于不需要离线消息的场景，资源占用少
2. Clean Session = false：适用于需要保持订阅和接收离线消息的场景
3. MQTT 5.0：使用 Clean Start 和 Session Expiry Interval 精确控制会话行为

## 8.4 异常处理

1. **断线重连**：实现连接丢失后的自动重连机制
2. **消息去重**：QoS 1 消息可能重复，业务侧需实现幂等处理
3. **超时处理**：合理设置连接超时和 Keepalive 间隔

## 8.5 性能优化

1. **批量发送**：减少频繁的小消息发送
2. **合理设置 Keepalive**：避免过短的心跳间隔增加网络负载
3. **使用共享订阅**：多个消费者均衡处理消息
4. **限制订阅数量**：单个客户端订阅过多主题会影响性能
5. **避免通配符滥用**：尽量使用精确主题匹配

全国统一服务热线

4008-555-800

金蝶天燕云计算股份有限公司(简称“金蝶天燕云”)成立于2000年，前身为“金蝶中间件公司”，是金蝶集团旗下新一代软件基础云平台服务商，云计算国家标准制定企业，国家信创产业核心软件企业。金蝶天燕是国家863重点研发计划与核高基重大专项承接企业，也是“两网一站四库十二金”国家重点工程的基础平台提供商，产品广泛应用于政府、军工、金融、能源等关键行业，累计服务客户总数超过10万家。

Apusic 金蝶天燕

云计算国家标准制定企业

金蝶集团旗下基础软件企业

信息技术应用创新核心企业

官网：www.apusic.com