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_rocketmq

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 RocketMQ（Apusic Distributed Message Queue for RocketMQ，简称：ADMQ for RocketMQ）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 RocketMQ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 RocketMQ产品相关的信息，请参阅以下ADMQ for RocketMQ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 RocketMQ 快速使用手册	简单介绍了如何快速上手使用ADMQ for RocketMQ 。
2	金蝶Apusic分布式消息队列for RocketMQ 安装手册	详细介绍如何在各操作系统上安装ADMQ for RocketMQ，以及ADMQ for RocketMQ服务启停等操作。
3	金蝶Apusic分布式消息队列for RocketMQ 消息引擎用户手册	详细介绍 ADMQ for RocketMQ 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 RocketMQ 管控台用户手册	详细介绍ADMQ for RocketMQ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 RocketMQ 开发手册	详细介绍基于各开发语言进行ADMQ for RocketMQ客户端应用开发的说明。
6	金蝶Apusic分布式消息队列for RocketMQ 迁移手册	详细介绍从RocketMQ迁移到ADMQ for RocketMQ的说明。
7	金蝶Apusic分布式消息队列for RocketMQ 运维手册	详细介绍ADMQ for RocketMQ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 RocketMQ 性能优化手册	详细介绍ADMQ for RocketMQ性能调优的说明。

1.3 技术支持

ADMQ for RocketMQ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 安装部署

2.1 部署启动

解压安装包

```
mkdir -p /apusic
cd /apusic
# 解压安装包
#tar zxvf ADMQ-V2.0.6-RocketMQ-构建日期.tar.gz -C /apusic/
tar zxvf ADMQ-V2.0.6.534-RocketMQ-20260625.tar.gz -C /apusic/
cd /apusic
# 建议nameserver broker组件文件夹结构分开
unzip rocketmq-V5.3.4-all.zip
mv rocketmq-V5.3.4-all nameserver
cd /apusic/nameserver

unzip rocketmq-V5.3.4-all.zip
mv rocketmq-V5.3.4-all broker
cd /apusic/broker
```

修改配置，修改对应配置文件的地址和端口

```
vi conf/nameserver.properties
vi conf/broker.conf
```

启动

```
bin/rocketmq start nameserver
bin/rocketmq start broker
```

2.2 连接参数

参数	说明	示例

namesrvAddr	NameServer 地址	localhost:9876
groupName	生产者/消费者组名	test-group
accessKey	ACL 访问密钥（可选）	test-user
secretKey	ACL 访问密钥（可选）	test-password

3 客户端开发

3.1 客户端依赖

3.1.1 Java

```
<dependency>
    <groupId>org.apache.rocketmq</groupId>
    <artifactId>rocketmq-client</artifactId>
    <version>4.9.8</version> (如果是5.x版本，可以用5.1.4)
</dependency>
```

3.1.2 Go

```
go get github.com/apache/rocketmq-clients/golang/v2
```

3.1.3 Python

```
pip install rocketmq-client-python
```

4 Java 开发示例

4.1 生产者

4.1.1 同步发送

```
import org.apache.rocketmq.client.producer.DefaultMQProducer;
import org.apache.rocketmq.client.producer.SendResult;
import org.apache.rocketmq.common.message.Message;

public class SyncProducer {
    public static void main(String[] args) throws Exception {
        DefaultMQProducer producer = new DefaultMQProducer("test-
group");
        producer.setNamesrvAddr("localhost:9876");
        producer.start();

        Message msg = new Message("test-topic", "tag-a", "test-key",
            "Hello ADMQ for RocketMQ!".getBytes("UTF-8"));
        SendResult result = producer.send(msg);
        System.out.println("SendResult: " + result);

        producer.shutdown();
    }
}
```

4.1.2 异步发送

```
import org.apache.rocketmq.client.producer.DefaultMQProducer;
import org.apache.rocketmq.client.producer.SendCallback;
import org.apache.rocketmq.client.producer.SendResult;
import org.apache.rocketmq.common.message.Message;

public class AsyncProducer {
```

```
    public static void main(String[] args) throws Exception {
        DefaultMQProducer producer = new DefaultMQProducer("test-
group");
        producer.setNamesrvAddr("localhost:9876");
        producer.start();

        Message msg = new Message("test-topic", "tag-a", "test-key",
            "Hello ADMQ for RocketMQ!".getBytes("UTF-8"));
        producer.send(msg, new SendCallback() {
            @Override
            public void onSuccess(SendResult sendResult) {
                System.out.println("Send success: " + sendResult);
            }

            @Override
            public void onException(Throwable e) {
                System.err.println("Send failed: " +
e.getMessage());
            }
        });

        Thread.sleep(5000);
        producer.shutdown();
    }
}
```

4.1.3 单向发送

```
Message msg = new Message("test-topic", "tag-a", "One-way
message".getBytes());
producer.sendOneway(msg);
```

4.2 消费者

4.2.1 push模式消费

```
import org.apache.rocketmq.client.consumer.DefaultMQPushConsumer;
import
org.apache.rocketmq.client.consumer.listener.ConsumeConcurrentlyContext
import
org.apache.rocketmq.client.consumer.listener.ConsumeConcurrentlyStatus;
import
org.apache.rocketmq.client.consumer.listener.MessageListenerConcurrentl
import org.apache.rocketmq.common.message.MessageExt;
import java.util.List;

public class PushConsumer {
    public static void main(String[] args) throws Exception {
        DefaultMQPushConsumer consumer = new
DefaultMQPushConsumer("test-group");
        consumer.setNamesrvAddr("localhost:9876");
        consumer.subscribe("test-topic", "*");

        consumer.registerMessageListener(new
MessageListenerConcurrently() {
            @Override
            public ConsumeConcurrentlyStatus
consumeMessage(List<MessageExt> msgs,
                    ConsumeConcurrentlyContext context) {
                for (MessageExt msg : msgs) {
                    System.out.println("Received: " + new
String(msg.getBody()));
                }
                return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
            }
        });

        consumer.start();
        System.out.println("Consumer started");
    }
}
```

4.2.2 pull模式消费

```
import org.apache.rocketmq.client.consumer.DefaultLitePullConsumer;
import org.apache.rocketmq.common.message.MessageExt;
import java.util.List;

public class PullConsumer {
    public static void main(String[] args) throws Exception {
        DefaultLitePullConsumer consumer = new
DefaultLitePullConsumer("test-group");
        consumer.setNamesrvAddr("localhost:9876");
        consumer.subscribe("test-topic", "*");
        consumer.start();

        while (true) {
            List<MessageExt> msgs = consumer.poll();
            for (MessageExt msg : msgs) {
                System.out.println("Received: " + new
String(msg.getBody()));
            }
        }
    }
}
```

5 Go 开发示例

5.1 生产者

```
package main

import (
    "context"
    "fmt"
    "github.com/apache/rocketmq-clients/golang/v2"
)

func main() {
    producer, err := golang.NewProducer(
        golang.WithEndpoints("localhost:9876"),
        golang.WithConsumerGroup("test-group"),
    )
    if err != nil {
        panic(err)
    }

    err = producer.Start()
    if err != nil {
        panic(err)
    }
    defer producer.Shutdown()

    msg := golang.NewMessage("test-topic", []byte("Hello ADMQ for
RocketMQ!"))
    msg.SetTag("tag-a")
    msg.SetKeys("test-key")

    resp, err := producer.Send(context.TODO(), msg)
    if err != nil {
```

```
        panic(err)
    }
    fmt.Printf("Message sent: %s\n", resp[0].MsgId)
}
```

5.2 消费者

```
package main

import (
    "context"
    "fmt"
    "github.com/apache/rocketmq-clients/golang/v2"
)

func main() {
    consumer, err := golang.NewSimpleConsumer(
        golang.WithEndpoints("localhost:9876"),
        golang.WithConsumerGroup("test-group"),

golang.WithSubscriptionExpressions(map[string]*golang.FilterExpression{
            "test-topic": golang.SUB_ALL,
        }),
    )
    if err != nil {
        panic(err)
    }

    err = consumer.Start()
    if err != nil {
        panic(err)
    }
    defer consumer.Shutdown()

    for {
```

```
        msgs, err := consumer.Receive(context.TODO(), 1, 30)
        if err != nil {
            fmt.Printf("Receive error: %v\n", err)
            continue
        }

        for _, msg := range msgs {
            fmt.Printf("Received: %s\n", string(msg.GetBody()))
            err = consumer.Ack(context.TODO(), msg)
            if err != nil {
                fmt.Printf("Ack error: %v\n", err)
            }
        }
    }
}
```

6 Python 开发示例

6.1 生产者

```python
from rocketmq.client import Producer, Message

producer = Producer('test-group')
producer.set_name_server_address('localhost:9876')
producer.start()

msg = Message('test-topic')
msg.set_tags('tag-a')
msg.set_keys('test-key')
msg.set_body('Hello ADMQ for RocketMQ!')

result = producer.send_sync(msg)
print('Message sent: {result.msg_id}')

producer.shutdown()
```

6.2 消费者

```python
from rocketmq.client import PushConsumer, ConsumeStatus

def callback(msg):
    print(f'Received: {msg.body.decode("utf-8")}')
    return ConsumeStatus.CONSUME_SUCCESS

consumer = PushConsumer('test-group')
consumer.set_name_server_address('localhost:9876')
consumer.subscribe('test-topic', '*', callback)
consumer.start()
```

```
print('Consumer started')
import time
time.sleep(60)
consumer.shutdown()
```

7 高级特性

7.1 顺序消息

```
// 发送顺序消息
Message msg = new Message("order-topic", "tag-a", "order-1",
"content".getBytes());
SendResult result = producer.send(msg, new MessageQueueSelector() {
    @Override
    public MessageQueue select(List<MessageQueue> mqs, Message msg,
Object arg) {
        Long orderId = (Long) arg;
        long index = orderId % mqs.size();
        return mqs.get((int) index);
    }
}, orderId);

// 消费顺序消息
consumer.registerMessageListener(new MessageListenerOrderly() {
    @Override
    public ConsumeOrderlyStatus consumeMessage(List<MessageExt>
msgs, ConsumeOrderlyContext context) {
        for (MessageExt msg : msgs) {
            System.out.println("Received: " + new
String(msg.getBody()));
        }
        return ConsumeOrderlyStatus.SUCCESS;
    }
});
```

7.2 事务消息

```
TransactionMQProducer producer = new TransactionMQProducer("tx-
group");
producer.setNamesrvAddr("localhost:9876");
producer.setTransactionListener(new TransactionListener() {
    @Override
    public LocalTransactionState executeLocalTransaction(Message
msg, Object arg) {
        try {
            // 执行本地事务
            executeLocalBusiness(msg);
            return LocalTransactionState.COMMIT_MESSAGE;
        } catch (Exception e) {
            return LocalTransactionState.ROLLBACK_MESSAGE;
        }
    }

    @Override
    public LocalTransactionState checkLocalTransaction(MessageExt
msg) {
        // 回查本地事务状态
        if (isTransactionSuccess(msg)) {
            return LocalTransactionState.COMMIT_MESSAGE;
        }
        return LocalTransactionState.ROLLBACK_MESSAGE;
    }
});
producer.start();
```

7.3 延时消息

```
Message msg = new Message("delay-topic", "tag-a",
"content".getBytes());
```

```
msg.setDelayTimeLevel(3);  // 延时 10 秒
producer.send(msg);
```

7.4 批量消费

```
consumer.setConsumeMessageBatchMaxSize(10);  // 每次消费 10 条消息
consumer.registerMessageListener(new MessageListenerConcurrently() {
    @Override
    public ConsumeConcurrentlyStatus consumeMessage(List<MessageExt>
msgs,
            ConsumeConcurrentlyContext context) {
        // 批量处理消息
        for (MessageExt msg : msgs) {
            processMessage(msg);
        }
        return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
    }
});
```

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