



APUSIC
固若长城
睿比世界

開發手冊

金蝶Apusic分散式訊息佇列V2.0.6_for_rocketmq

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 技術支援
- 2 安裝部署
 - 2.1 部署啟動
 - 2.2 連線參數
- 3 用戶端開發
 - 3.1 用戶端依賴
 - 3.1.1 Java
 - 3.1.2 Go
 - 3.1.3 Python
- 4 Java 開發範例
 - 4.1 生產者
 - 4.1.1 同步發送
 - 4.1.2 非同步發送
 - 4.1.3 單向發送
 - 4.2 消費者
 - 4.2.1 push模式消費
 - 4.2.2 pull模式消費
- 5 Go 開發範例
 - 5.1 生產者
 - 5.2 消費者
- 6 Python 開發範例
 - 6.1 生產者
 - 6.2 消費者
- 7 進階特性
 - 7.1 順序訊息
 - 7.2 事務訊息
 - 7.3 延時訊息
 - 7.4 批量消費

1 前言

本文檔為金蝶Apusic分散式訊息佇列for RocketMQ（Apusic Distributed Message Queue for RocketMQ，簡稱：ADMQ for RocketMQ）的開發使用說明，幫助用戶快速學習如何使用金蝶Apusic分散式訊息佇列for RocketMQ進行開發。

1.1 適用對象

本文檔適用於IT資訊化業務負責人、研發經理、軟體專案經理、軟體架構師、運維工程師。

1.2 相關文檔

了解更多ADMQ for RocketMQ產品相關的資訊，請參閱以下ADMQ for RocketMQ產品手冊文檔集：

序號	手冊文檔	說明
1	金蝶Apusic分散式訊息佇列for RocketMQ 快速使用手冊	簡單介紹了如何快速上手使用ADMQ for RocketMQ。
2	金蝶Apusic分散式訊息佇列for RocketMQ 安裝手冊	詳細介紹如何在各作業系統上安裝ADMQ for RocketMQ，以及ADMQ for RocketMQ服務啟停等操作。
3	金蝶Apusic分散式訊息佇列for RocketMQ 訊息引擎使用者手冊	詳細介紹 ADMQ for RocketMQ 訊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分散式訊息佇列for RocketMQ 管控台使用者手冊	詳細介紹ADMQ for RocketMQ管控台相關功能的使用和操作說明。
5	金蝶Apusic分散式訊息佇列for RocketMQ 開發手冊	詳細介紹基於各開發語言進行ADMQ for RocketMQ用戶端應用開發的說明。
6	金蝶Apusic分散式訊息佇列for RocketMQ 遷移手冊	詳細介紹從RocketMQ遷移到ADMQ for RocketMQ的說明。
7	金蝶Apusic分散式訊息佇列for RocketMQ 運維手冊	詳細介紹ADMQ for RocketMQ的監控、運維、安全加固等運維說明。
8	金蝶Apusic分散式訊息佇列for RocketMQ 效能優化手冊	詳細介紹ADMQ for RocketMQ效能調優的說明。

1.3 技術支援

ADMQ for RocketMQ產品提供全面的技術支援服務，您可以透過以下方式獲得技術支援：

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

您在取得技術支援時，請提供如下資訊：

1. 您的姓名
2. 公司資訊與聯絡方式
3. 作業系統及其版本
4. 產品版本號
5. 出現異常及錯誤的日誌、截圖等詳細資訊

2 安裝部署

2.1 部署啟動

解壓安裝包

```
mkdir -p /apusic
cd /apusic
# 解壓安裝包
#tar zxvf ADMQ-V2.0.6-RocketMQ-構建日期.tar.gz -C /apusic/
tar zxvf ADMQ-V2.0.6.534-RocketMQ-20260625.tar.gz -C /apusic/
cd /apusic
# 建議nameserver broker組件資料夾結構分開
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，修改對應配置文件的位址和連接埠

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

啟動

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

2.2 連線參數

參數	說明	範例
----	----	----

namesrvAddr	NameServer 位址	localhost:9876
groupName	生產者/消費者組名	test-group
accessKey	ACL 存取金鑰（可選）	test-user
secretKey	ACL 存取金鑰（可選）	test-password

3 用戶端開發

3.1 用戶端依賴

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 開發範例

4.1 生產者

4.1.1 同步發送

```
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 非同步發送

```
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 單向發送

```

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

```

4.2 消費者

4.2.1 push模式消費

```

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.MessageListenerConcurrently;
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模式消費

```
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 開發範例

5.1 生產者

```
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 消費者

```

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 開發範例

6.1 生產者

```
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 消費者

```
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 進階特性

7.1 順序訊息

```
// 發送順序訊息
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);

// 消費順序訊息
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 事務訊息

```

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

    @Override
    public LocalTransactionState checkLocalTransaction(MessageExt
msg) {
        // 回查本地事務狀態
        if (isTransactionSuccess(msg)) {
            return LocalTransactionState.COMMIT_MESSAGE;
        }
        return LocalTransactionState.ROLLBACK_MESSAGE;
    }
});
producer.start();

```

7.3 延時訊息

```

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

```

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

7.4 批量消費

```
consumer.setConsumeMessageBatchMaxSize(10); // 每次消費 10 條訊息
consumer.registerMessageListener(new MessageListenerConcurrently() {
    @Override
    public ConsumeConcurrentlyStatus consumeMessage(List<MessageExt>
msgs,
        ConsumeConcurrentlyContext context) {
        // 批量處理訊息
        for (MessageExt msg : msgs) {
            processMessage(msg);
        }
        return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
    }
});
```

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

