



Development Manual

Kingdee Apusic Distributed Message Queue
V2.0.6_for_rocketmq

Copyright Notice

All intellectual property rights including software copyrights involved in this document have been legally registered and duly owned by Kingdee Apusic Cloud Computing Co., Ltd. Such rights are protected by Copyright Law of the People's Republic of China, Regulations on Protection of Computer Software, Regulations on Intellectual Property Protection, relevant international copyright conventions, laws, regulations and other intellectual property legislations. Unauthorized use is strictly prohibited.

Disclaimer

All copyright information contained herein is legally owned by Kingdee Apusic Cloud Computing Co., Ltd. and protected by applicable laws. The Company shall bear no liability for any information unrelated to Kingdee Apusic contained in this document. You may view, copy and print this document only within the scope permitted by Copyright Law of the People's Republic of China. No entity or individual may use, modify or republish any part or content of this document without written authorization from Kingdee Apusic Cloud Computing Co., Ltd. Any violation shall constitute infringement, and the Company reserves the right to pursue legal liabilities accordingly.

Updates to this document shall not be notified separately. You may consult the Company regarding issues related to this document. All rights not expressly granted by the Company are reserved.

Trademark Notice

 is a registered trademark filed with the National Intellectual Property Administration of PRC by Kingdee Apusic Cloud Computing Co., Ltd. Shenzhen. Exclusive rights to the registered trademark are legally owned by Kingdee Apusic and protected by law. Without written consent of Kingdee Apusic, no entity or individual shall use, copy, modify, disseminate, transcribe, or bundle and sell any part of this trademark with other products in any manner or for any reason. Kingdee Apusic shall pursue legal liabilities against any infringement of its trademark rights. All other trademarks or registered trademarks mentioned in this document belong to their respective owners.

Contents

- 1 Preface
 - 1.1 Target Audience
 - 1.2 Related Documents
 - 1.3 Technical Support
- 2 Installation and Deployment
 - 2.1 Deployment and Startup
 - 2.2 Connection Parameters
- 3 Client Development
 - 3.1 Client Dependencies
 - 3.1.1 Java
 - 3.1.2 Go
 - 3.1.3 Python
- 4 Java Development Examples
 - 4.1 Producer
 - 4.1.1 Synchronous Send
 - 4.1.2 Asynchronous Send
 - 4.1.3 One-way Send
 - 4.2 Consumer
 - 4.2.1 Push Mode Consumption
 - 4.2.2 Pull Mode Consumption
- 5 Go Development Examples
 - 5.1 Producer
 - 5.2 Consumer
- 6 Python Development Examples
 - 6.1 Producer
 - 6.2 Consumer
- 7 Advanced Features
 - 7.1 Ordered Messages
 - 7.2 Transactional Messages
 - 7.3 Delayed Messages
 - 7.4 Batch Consumption

1 Preface

This document is the development guide for Kingdee Apusic Distributed Message Queue for RocketMQ (Apusic Distributed Message Queue for RocketMQ, referred to as: ADMQ for RocketMQ), helping users quickly learn how to use ADMQ for RocketMQ for development.

1.1 Target Audience

This document is intended for IT information business leaders, R&D managers, software project managers, software architects, and operations engineers.

1.2 Related Documents

For more information about ADMQ for RocketMQ products, please refer to the following ADMQ for RocketMQ product documentation set:

No.	Document	Description
1	Kingdee Apusic Distributed Message Queue for RocketMQ Quick Start Guide	Briefly introduces how to quickly get started with ADMQ for RocketMQ.
2	Kingdee Apusic Distributed Message Queue for RocketMQ Installation Guide	Details how to install ADMQ for RocketMQ on various operating systems, as well as ADMQ for RocketMQ service start/stop operations.
3	Kingdee Apusic Distributed Message Queue for RocketMQ Message Engine User Guide	Details the usage, configuration, management, and supporting tools of ADMQ for RocketMQ message engine related features.
4	Kingdee Apusic Distributed Message Queue for RocketMQ Console User Guide	Details the usage and operation instructions of the ADMQ for RocketMQ console.
5	Kingdee Apusic Distributed Message Queue for RocketMQ Development Guide	Details the instructions for developing ADMQ for RocketMQ client applications in various development languages.
6	Kingdee Apusic Distributed Message Queue for RocketMQ Migration Guide	Details the instructions for migrating from RocketMQ to ADMQ for RocketMQ.
7	Kingdee Apusic Distributed Message Queue for RocketMQ Operations Guide	Details ADMQ for RocketMQ monitoring, operations, security hardening and other operations instructions.

8	Kingdee Apusic Distributed Message Queue for RocketMQ Performance Optimization Guide	Details the instructions for ADMQ for RocketMQ performance tuning.
---	--	--

1.3 Technical Support

ADMQ for RocketMQ provides comprehensive technical support services. You can obtain technical support through the following methods:

- Website: www.apusic.com
- Phone: 400-855-5800
- Email: support@apusic.com
- Kingdee Cloud Community: <https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN>

When contacting technical support, please provide the following information:

1. Your name
2. Company information and contact details
3. Operating system and its version
4. Product version number
5. Detailed information such as logs, screenshots of exceptions and errors

2 Installation and Deployment

2.1 Deployment and Startup

Extract the installation package

```
mkdir -p /apusic
cd /apusic
# Extract the installation package
#tar zxvf ADMQ-V2.0.6-RocketMQ-{build_date}.tar.gz -C /apusic/
tar zxvf ADMQ-V2.0.6.534-RocketMQ-20260625.tar.gz -C /apusic/
cd /apusic
# It is recommended to separate the nameserver and broker component
folder structures
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
```

Modify the configuration, update the address and port in the corresponding configuration file

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

Start

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

2.2 Connection Parameters

Parameter	Description	Example
namesrvAddr	NameServer address	localhost:9876
groupName	Producer/Consumer group name	test-group
accessKey	ACL access key (optional)	test-user
secretKey	ACL secret key (optional)	test-password

3 Client Development

3.1 Client Dependencies

3.1.1 Java

```
<dependency>  
  <groupId>org.apache.rocketmq</groupId>  
  <artifactId>rocketmq-client</artifactId>  
  <version>4.9.8</version> (For 5.x version, use 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 Development Examples

4.1 Producer

4.1.1 Synchronous Send

```
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 Asynchronous Send

```
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 One-way Send

```

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

```

4.2 Consumer

4.2.1 Push Mode Consumption

```

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 Mode Consumption

```
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 Development Examples

5.1 Producer

```
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 Consumer

```

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 Development Examples

6.1 Producer

```
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 Consumer

```
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 Advanced Features

7.1 Ordered Messages

```
// Send ordered message
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);

// Consume ordered message
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 Transactional Messages

```

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

    @Override
    public LocalTransactionState checkLocalTransaction(MessageExt
msg) {
        // Check back local transaction status
        if (isTransactionSuccess(msg)) {
            return LocalTransactionState.COMMIT_MESSAGE;
        }
        return LocalTransactionState.ROLLBACK_MESSAGE;
    }
});
producer.start();

```

7.3 Delayed Messages

```

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

```

```
msg.setDelayTimeLevel(3); // Delay for 10 seconds
producer.send(msg);
```

7.4 Batch Consumption

```
consumer.setConsumeMessageBatchMaxSize(10); // Consume 10 messages
at a time
consumer.registerMessageListener(new MessageListenerConcurrently() {
    @Override
    public ConsumeConcurrentlyStatus consumeMessage(List<MessageExt>
msgs,
        ConsumeConcurrentlyContext context) {
        // Process messages in batch
        for (MessageExt msg : msgs) {
            processMessage(msg);
        }
        return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
    }
});
```

UNIFIED SERVICE HOTLINE

4008-555-800

Kingdee Apusic Cloud Computing Co., Ltd. (referred to as “Kingdee Apusic”) was founded in 2000. Its predecessor was “Kingdee Middleware Company.” It is a basic software platform provider under Kingdee Group, a national standards-setting enterprise for cloud computing, and a core software enterprise in the national IT application innovation industry. Kingdee Apusic is an enterprise undertaking the National 863 Key R&D Program and the National Major Science and Technology Project for Core Electronic Components, High-end General-purpose Chips and Fundamental Software, and is also a basic platform provider for the national key projects of “Two Networks, One Website, Four Databases, and Twelve Golden Projects.” Its products are widely used in key industries such as government, military, finance, and energy, and it has cumulatively served more than 100,000 customers.



National standard setter for cloud computing
Basic software enterprise under Kingdee Group
Core enterprise for IT application innovation
Website: www.apusic.com

