



Quick Start

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 Intended Audience
 - 1.2 Related Documents
 - 1.3 Technical Support
- 2 Quick Deployment
 - 2.1 Environment Preparation
 - 2.2 One-Click Deployment
 - 2.3 Console Deployment
- 3 Quick Verification
 - 3.1 Access Console
 - 3.2 Create Test Resources
 - 3.3 Sending and Receiving Messages
 - 3.3.1 Using Command Line Tools
 - 3.3.2 Using Console to Send Messages
 - 3.3.3 Using Open Source SDK for Message Sending and Receiving (Java SDK Example)
 - 3.3.3.1 Overview
 - 3.3.3.2 Producing Messages
 - 3.3.3.3 Consuming Messages (Using Remoting Protocol as an Example)
- 4 Quick Client Integration
 - 4.1 Java Client Example
 - 4.2 Go Client Example

1 Preface

This document is the quick start manual for Kingdee Apusic Distributed Message Queue for RocketMQ (Apusic Distributed Message Queue for RocketMQ, abbreviation: ADMQ for RocketMQ). It provides product introduction, basic steps for installing and deploying the product, and verification methods.

1.1 Intended 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 Manual	A brief introduction on how to quickly get started with ADMQ for RocketMQ.
2	Kingdee Apusic Distributed Message Queue for RocketMQ Installation Manual	Detailed instructions on installing ADMQ for RocketMQ on various operating systems, as well as starting and stopping ADMQ for RocketMQ services.
3	Kingdee Apusic Distributed Message Queue for RocketMQ Message Engine User Manual	Detailed description of the usage, configuration, management, and supporting tools of the ADMQ for RocketMQ message engine.
4	Kingdee Apusic Distributed Message Queue for RocketMQ Console User Manual	Detailed description of the usage and operation instructions for the ADMQ for RocketMQ console.
5	Kingdee Apusic Distributed Message Queue for RocketMQ Development Manual	Detailed instructions for developing ADMQ for RocketMQ client applications in various programming languages.
6	Kingdee Apusic Distributed Message Queue for RocketMQ Migration Manual	Detailed instructions on migrating from RocketMQ to ADMQ for RocketMQ.
7	Kingdee Apusic Distributed Message Queue for RocketMQ Operations Manual	Detailed description of ADMQ for RocketMQ monitoring, operations, security hardening, and other O&M instructions.
8	Kingdee Apusic Distributed Message Queue for RocketMQ Performance Optimization Manual	Detailed description of 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 channels:

- 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 and screenshots of anomalies and errors

2 Quick Deployment

2.1 Environment Preparation

Component	Minimum Requirement
CPU	4 cores
Memory	16 GB
Disk	200 GB SSD
Operating System	CentOS 7+ / Ubuntu 18.04+

2.2 One-Click Deployment

```
mkdir -p /apusic
cd /apusic
# 1. Extract installation package
#tar zxvf ADMQ-V2.0.6-RocketMQ-<build-date>.tar.gz -C /apusic/
tar zxvf ADMQ-V2.0.6.534-RocketMQ-20260624.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

# 2. Start NameServer
bin/rocketmq start nameserver

# 3. Start Broker
bin/rocketmq start broker
```

```
# 4. Verify status  
bin/mqadmin clusterList -n localhost:9876
```

2.3 Console Deployment

```
# 1. Extract console  
tar zxvf admq-manager-V2.0.6.tar.gz -C /apusic/  
cd /apusic/admq-manager-V2.0.6  
  
# 2. Start console  
bin/admq-service start
```

3 Quick Verification

3.1 Access Console

```
Address: http://<Console IP>:12305;https://<Console IP>:12306
Default account: admq / 11111111
```

3.2 Create Test Resources

1. Create Topic

- Go to the "Resource Management" - "Topic Management" page
- Click "New", enter the topic name (e.g., test-topic)
- Configure the number of read/write queues (e.g., 3)
- Confirm creation

2. Create Subscription Group

- Go to the "Resource Management" - "Subscription Group Management" page
- Click "New", enter the subscription group name (e.g., group)
- Configure subscription group information
- Confirm creation

3. View Message Sending

- Go to the "Message Query" page
- Select topic: test-topic
- You can view messages of this Topic

3.3 Sending and Receiving Messages

3.3.1 Using Command Line Tools

```
# Send message
bin/mqadmin sendMessage -n localhost:9876 -t test-topic -p "Hello ADMQ for
RocketMQ!"

# View message
bin/mqadmin queryMsgByKey -n localhost:9876 -t test-topic -k test-key
```

3.3.2 Using Console to Send Messages

1. Go to the "Resource Management" - "Topic Management" page
2. Select test-topic, click "Send Message"
3. Fill in the message content, Tag, Key, and other information
4. Click Send

3.3.3 Using Open Source SDK for Message Sending and Receiving (Java SDK Example)

3.3.3.1 Overview

Currently, RocketMQ is compatible with open-source features. Based on differences in underlying communication protocols, it mainly supports two series of client SDKs: the Remoting protocol and the gRPC protocol. For a comparison between the Remoting protocol SDK and the gRPC protocol SDK, please refer to the following:

Comparison Item	Remoting Protocol SDK	gRPC Protocol SDK
Multi-language Support	Primarily Java, other languages implemented by third-party repositories	Java/C/C++/.NET/Go/Rust
Interface Scope	Producer PushConsumer PullConsumer LitePullConsumer Admin	Producer PushConsumer(Java only) SimpleConsumer
Compatible Versions	Compatible with 4.x and 5.x version servers	Only supports 5.x version servers

3.3.3.2 Producing Messages

Create a message producer

```
// Instantiate the message producer Producer
DefaultMQProducer producer = new DefaultMQProducer(
    namespace,
    groupName,
    new AclClientRPCHook(new SessionCredentials(accessKey, secretKey)) //
    ACL permissions
);
// Set the NameServer address
producer.setNamesrvAddr(nameserver);
// Start the Producer instance
producer.start();
```

Sending messages There are multiple ways to send messages: synchronous sending, asynchronous sending, one-way sending, etc.

- Synchronous messages

```
for (int i = 0; i < 10; i++) {
    // Create a message instance, set topic and message content
    Message msg = new Message(topic_name, "TAG", ("Hello RocketMQ " +
i).getBytes(RemotingHelper.DEFAULT_CHARSET));
    // Send message
    SendResult sendResult = producer.send(msg);
    System.out.printf("%s\n", sendResult);
}
```

- Asynchronous messages

```
// Set not to retry after sending failure
producer.setRetryTimesWhenSendAsyncFailed(0);
// Set the number of messages to send
int messageCount = 10;
final CountDownLatch countDownLatch = new CountDownLatch(messageCount);
for (int i = 0; i < messageCount; i++) {
    try {
        final int index = i;
        // Create message entity, set topic and message content
        Message msg = new Message(topic_name, "TAG", ("Hello rocketMq " +
index).getBytes(RemotingHelper.DEFAULT_CHARSET));
        producer.send(msg, new SendCallback() {
            @Override
            public void onSuccess(SendResult sendResult) {
                // Message sent successfully logic
                countDownLatch.countDown();
                System.out.printf("%-10d OK %s %n", index,
sendResult.getMsgId());
            }

            @Override
            public void onException(Throwable e) {
                // Message sending failed logic
                countDownLatch.countDown();
            }
        });
    } catch (Exception e) {
        // Handle exception
    }
}
```

```

        System.out.printf("%-10d Exception %s %n", index,
e);

        e.printStackTrace();

    }

    });
} catch (Exception e) {
    e.printStackTrace();
}
}

countDownLatch.await(5, TimeUnit.SECONDS);

```

- One-way sending

```

for (int i = 0; i < 10; i++) {
    // Create a message instance, set topic and message content
    Message msg = new Message(topic_name, "TAG", ("Hello RocketMQ " +
i).getBytes(RemotingHelper.DEFAULT_CHARSET));
    // Send one-way message
    producer.sendOneway(msg);
}

```

3.3.3.3 Consuming Messages (Using Remoting Protocol as an Example)

Create a consumer Supports two consumption modes: push and pull

- Push consumer

```

// Instantiate the consumer
DefaultMQPushConsumer pushConsumer = new DefaultMQPushConsumer(
    namespace,
    groupName,
    new AclClientRPCHook(new SessionCredentials(accessKey, secretKey))); //ACL
permissions
// Set the NameServer address
pushConsumer.setNamesrvAddr(nameserver);

```

- Pull consumer

```

// Instantiate the consumer
DefaultLitePullConsumer pullConsumer = new DefaultLitePullConsumer(
    namespace,

```

```

groupName,
    new AclClientRPCHook(new SessionCredentials(accessKey, secretKey)));
// Set the NameServer address
pullConsumer.setNamesrvAddr(nameserver);
// Set to start consumption from the first offset
pullConsumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME_FROM_FIRST_OFFSET);

```

Subscribing to messages The subscription method differs depending on the consumption mode.

- Push subscription

```

// Subscribe to topic
pushConsumer.subscribe(topic_name, "*");
// Register callback implementation class to process messages pulled from
broker
pushConsumer.registerMessageListener((MessageListenerConcurrently) (msgs,
context) -> {
    // Message processing logic
    System.out.printf("%s Receive New Messages: %s %n",
Thread.currentThread().getName(), msgs);
    // Mark the message as successfully consumed, return processing status
based on consumption result
    return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
});
// Start the consumer instance
pushConsumer.start();

```

- Pull subscription

```

// Subscribe to topic
pullConsumer.subscribe(topic_name, "*");
// Start the consumer instance
pullConsumer.start();
try {
    System.out.printf("Consumer Started.%n");
    while (true) {
        // Pull messages
        List<MessageExt> messageExts = pullConsumer.poll();
        System.out.printf("%s%n", messageExts);
    }
}

```

```
} finally {  
    pullConsumer.shutdown();  
}
```

4 Quick Client Integration

4.1 Java Client Example

```
import org.apache.rocketmq.client.producer.DefaultMQProducer;
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.Message;
import org.apache.rocketmq.common.message.MessageExt;
import java.util.List;

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

        // Send message
        Message msg = new Message("test-topic", "tag-a", "test-key",
            "Hello ADMQ for RocketMQ!".getBytes("UTF-8"));
        producer.send(msg);
        System.out.println("Message sent");
        producer.shutdown();

        // Consumer
        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 Go Client Example

```

package main

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

func main() {
    // Producer
    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!"))  
resp, err := producer.Send(context.TODO(), msg)  
if err != nil {  
    panic(err)  
}  
fmt.Printf("Message sent: %s\n", resp[0].MsgId)  
}
```

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

